mi COMPUTER

34

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona, y comercializado en exclusiva por Distribuidora Olimpia, S.A., Barcelona

Volumen III - Fascículo 34

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas, F. Blasco

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, Barcelona-8
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-85822-94-3 (tomo 3)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 058409
Impreso en España - Printed in Spain - Septiembre 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, Madrid-34.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio Blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Ferrenquín a Cruz de Candelaria, 178, Caracas, y todas sus sucursales en el interior del país.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 3371872 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Distribuidora Olimpia (Paseo de Gracia, 88, 5.°, Barcelona-8), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Distribuidora Olimpia, en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.





Trabajo en la sombra

Los ordenadores cuentan con un programa incorporado que, en silencio, ejecuta las rutinas de funcionamiento de la máquina



Paul Chave

El "administrador"
Todos los ordenadores poseen alguna forma de sistema operativo, un programa que administra el funcionamiento del ordenador y controla todos los dispositivos conectados al sistema

El sistema operativo es un componente básico de todo ordenador, porque constituye un vínculo entre el hardware y el software. Sin embargo, como los sistemas operativos realizan la mayoría de sus tareas inadvertidamente, para muchas personas, en particular los usuarios de ordenadores personales, su labor es casi ignorada. Por este motivo es necesario presentar un panorama de ellos.

Los lenguajes de alto nivel permiten que el programador quede aislado del funcionamiento interno de la CPU y posibilitan una mayor portabilidad de programas entre un sistema y otro. Siempre y cuando un lenguaje esté razonablemente estandarizado, las instrucciones deberían funcionar en cualquier máquina que dispusiera del lenguaje. El intérprete o el compilador que procese el código fuente del lenguaje de alto nivel se encarga de los detalles de distribución de la memoria y demás. El intérprete o compilador de alto nivel también es un programa y se debe cargar en la memoria principal para que pueda así convertir el código fuente de alto nivel a instrucciones en código objeto listas para su ejecución. Los distintos BASIC en ROM ya están presentes, de forma permanente, en la memoria, y están listos para ser utilizados tan pronto como se conecte la máquina.

Sin embargo, para que un ordenador funcione no basta con disponer de un programa dentro de él que sea capaz de convertir código fuente en lenguaje máquina. Es necesario que haya aún otro programa funcionando en las sombras que se ocupe de la "administración doméstica". Como ejemplo de esta administración interna, consideremos el problema de conseguir que una letra digitada en el teclado aparezca en la pantalla. En algún lugar de la memoria ha de haber un programa que le diga constantemente a la CPU que verifique el teclado para ver si se ha pulsado alguna tecla. Si se hubiera pulsado alguna, el programa debería determinar de qué tecla se trata y luego instruir al sistema de circuitos de video para que produzca el patrón de puntos correcto, en la secuencia apropiada, para la salida a la pantalla. Se dice que las actividades como ésta son "transparentes" para el usuario.

Del mismo modo, cuando se emite una orden como CSAVE para guardar un archivo en cassette, el programador no se ha de preocupar por cómo se convierten los datos a la forma adecuada para el almacenamiento en cassette: todo forma parte del sistema operativo.

El sistema operativo es el *programa background*, es decir, de baja prioridad que se ejecuta constantemente y que supervisa todo lo demás. Cuando el ordenador en cuestión es un pequeño sistema informático basado en ROM con un BASIC incorporado, surge una pequeña fuente de confusión, porque con frecuencia el BASIC y el sistema operativo están retenidos en la misma ROM. En su forma más sen-

cilla, entonces, una ROM interna contendrá todo el software necesario para que el sistema funcione, aparte de los programas para aplicaciones (juegos, tratamiento de textos, etc.) cargados o escritos por el usuario. Parte de esta ROM contendrá el código necesario para convertir los programas de aplicaciones escritos en BASIC a lenguaje máquina (el *intérprete*); otra parte contendrá el código necesario para introducir y modificar los programas escritos por el usuario (el *editor*); y otra parte contendrá el software de administración interna necesario para cuidar del teclado, visualizar caracteres y gráficos, aceptar datos provenientes de cassettes y destinarlos a las partes pertinentes de la memoria, etc. (el *monitor*).

El término *monitor*, que no se debe confundir con un televisor o un monitor para visualización, es un sinónimo aproximado de "sistema operativo". En su forma más sencilla, el monitor puede hacer apenas poco más que aceptar instrucciones en lenguaje máquina, colocarlas en la posición de memoria adecuada y supervisar su ejecución. Cuando la administración interna se convierte en algo un poco más avanzado que esto, se tiende a aludir al propio monitor como al sistema operativo.

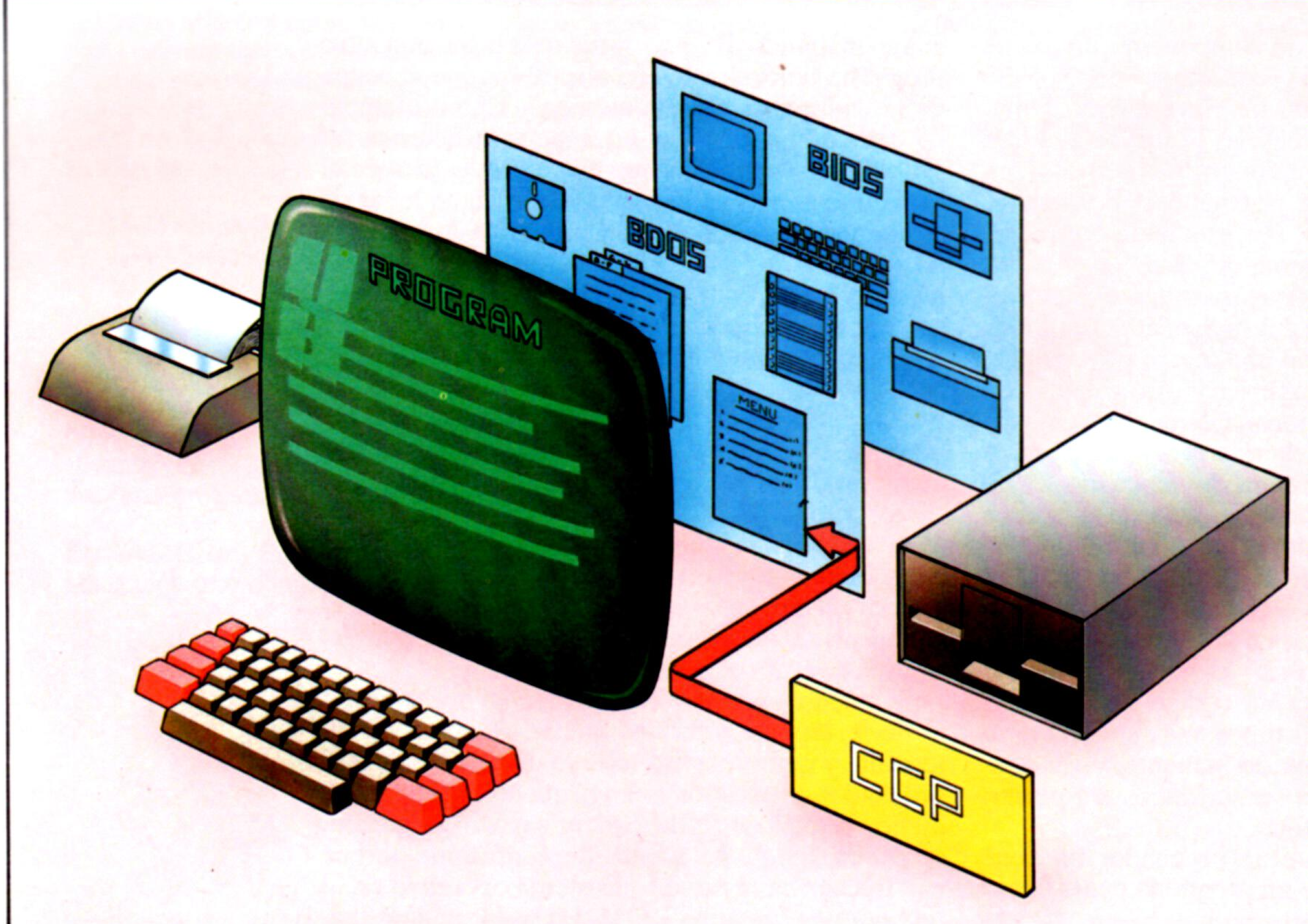
En el otro extremo están los ordenadores basados en disco, que a menudo se utilizan en oficinas como pequeños sistemas de gestión, y que poseen poderosos sistemas operativos. Antes de considerar ordenadores intermedios, como el Apple, vamos a analizar el tipo de sistema operativo que necesita un sistema sólo de disco.

Un sistema informático basado íntegramente en discos flexibles para su software, por lo general, tendrá muy poco almacenado en ROM con carácter permanente, aparte de un cargador *bootstrap* y algunas rutinas de administración interna. Cuando se conecta un ordenador de este tipo, el cargador *bootstrap* sólo contiene el código de lenguaje máquina suficiente para indicarle a la CPU cómo acceder a una unidad de disco y cargar el sistema operativo en la memoria principal.

El sistema operativo cargado en RAM ha de ser capaz de hacer más que los sistemas operativos de sistemas más convencionales basados en ROM. Se convierte en lo que denomina un *DOS* o *sistema operativo en disco*. Un sistema operativo como éste amplía sus funciones normales de administración interna mediante la adición de órdenes que actúan directamente en los archivos almacenados en el disco. Un sistema operativo se espera que incluya órdenes para listar los nombres de los archivos almacenados en el disco, borrar nombres de archivo o cambiarlos por otros, y copiar archivos de disco en la memoria principal o en otros discos.

Los sistemas operativos de sistemas más sencillos, basados en ROM, normalmente no poseen órdenes para manipulación de archivos tan sofisticadas y podrían no tener más que una orden simple para cargar un archivo con nombre desde cinta o para almacenar un archivo en cinta bajo un nombre determinado. Un sistema operativo sofisticado sabrá la dirección exacta del disco o cinta donde se encuentra almacenado cualquier archivo. Los sistemas operativos menos avanzados sólo son capaces de explorar en todos los archivos presentes hasta hallar el del nombre buscado, y después cargarlo: o escribir un archivo bajo un nombre de archivo dado en la cinta en el punto, cualquiera que fuera, en donde se encontrara la cinta en el momento de emitirse la orden.

Un buen ejemplo de sistema informático intermedio entre los dos anteriores es el BBC Micro, que es igualmente eficaz en la manipulación de archivos en cassette o en disco, utilizando en gran parte las mismas órdenes. El sistema operativo resi-



Controles de operación

El CP/M está dispuesto en capas para posibilitar que tanto él mismo como los programas para él escritos se trasladen fácilmente de un micro a otro. El programa del usuario funciona a través del BDOS (sistema operativo en disco básico), conjunto de rutinas que proporcionan funciones estándar y facilidades para el programa. El BDOS es el mismo en todas las máquinas, de modo que un programa que lo utilice funcionará en cualquier otro sistema CP/M. Cuando el BDOS necesita realizar una tarea determinada, llama al BIOS (sistema básico de entrada-salida). Éste se adapta especialmente a cada máquina y es el vínculo de software entre el BDOS y los componentes físicos del ordenador. El CP/M también incluye el CCP (*Command Control Program*: programa de control de órdenes), que permite que el usuario dé entrada a órdenes para ejecutar programas, cambiar el nombre de archivos y borrarlos, y, en general, controlar el sistema informático

Kevin Jones



de en una ROM, pero se trata de una ROM físicamente separada y se puede pensar en ella como un sistema operativo que se ha cargado en la memoria desde un dispositivo de memoria externo. Sus funciones incluyen órdenes para manipulación de archivos considerablemente más avanzadas que aquellas de que disponen otros ordenadores con ROM solamente, como el Spectrum.

El sistema operativo que utiliza un ordenador se puede, entonces, considerar como un programa que posee la función de situarse entre el usuario y el resto del sistema informático, incluyendo su CPU, su software de sistemas (como los lenguajes de programación) y el software de aplicaciones.

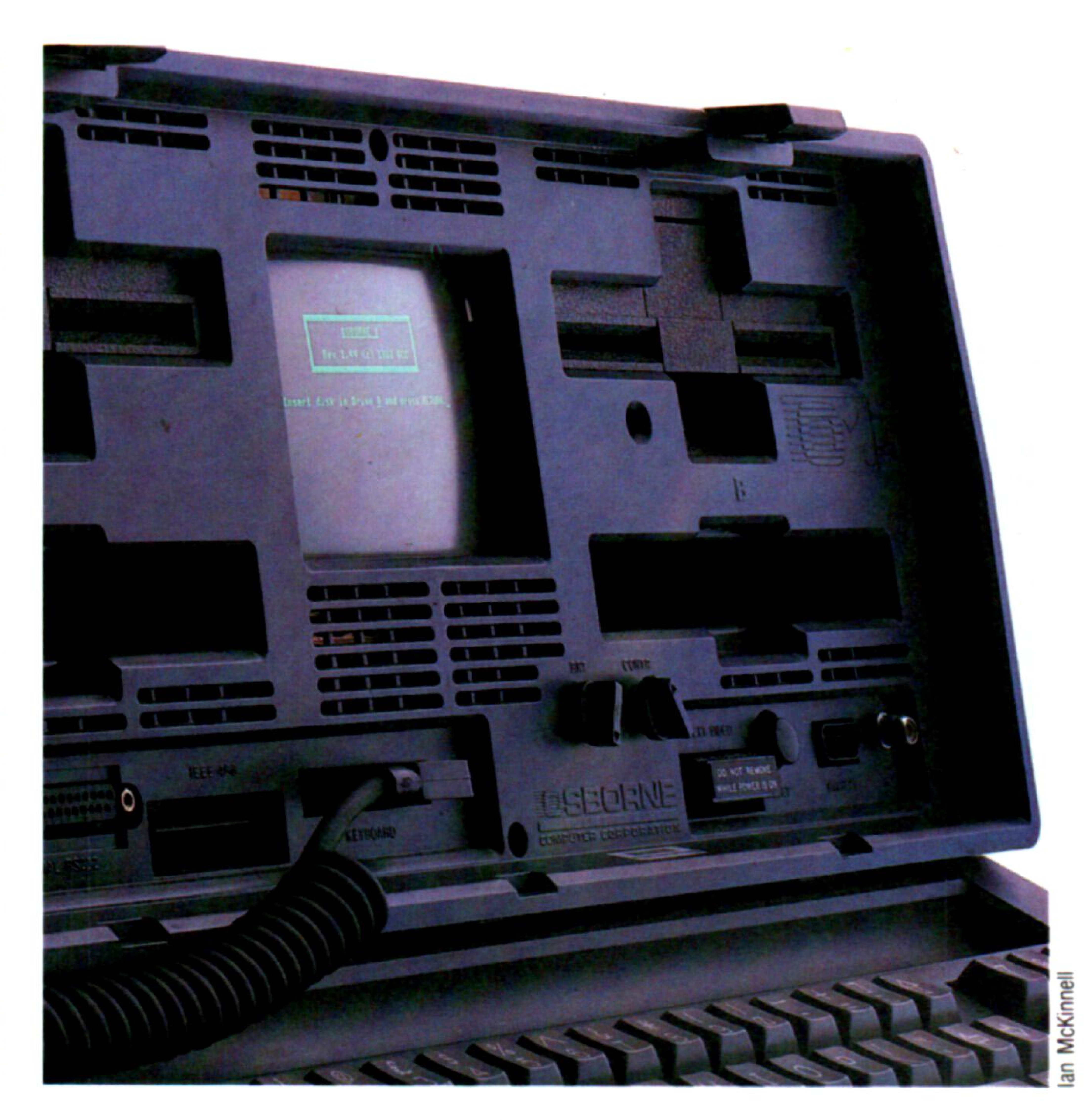
Compatibilidad

La capacidad de utilizar el software en más de un sistema de ordenador se conoce por compatibilidad. En términos amplios, ésta tiene dos aspectos. El primero de ellos es el hecho de que procesadores diferentes exigen juegos de instrucciones diferentes para realizar operaciones equivalentes. Por consiguiente, las instrucciones en lenguaje máquina para, supongamos, sumar los contenidos de dos posiciones de memoria, tendrían una forma si se escribieran para el 6502 (utilizado en el Apple) y otra forma totalmente distinta si se requiriera la misma operación en un ordenador Z80 como el Spectrum. El problema que entraña convertir código de alto nivel a código de lenguaje máquina idóneo es, no obstante, responsabilidad del intérprete o el compilador empleados. Para cada CPU diferente se han de escribir intérpretes y compiladores diferentes.

Existe, no obstante, un problema separado que afecta a la compatibilidad del software. Aun cuando se utilice la misma CPU, como en el Apple y el BBC, existen otras complicaciones. Se utilizan distintas posiciones de memoria para la memoria de video, se necesitan diferentes códigos para desplazar el cursor por la pantalla, se proporcionan distintos recursos de entrada y salida, y de este modo sucesivamente.

Para superar este problema se desarrollaron sistemas operativos en disco genéricos que permitieran que todo el software escrito, por ejemplo, para un ordenador Z80 basado en disco, se pudiera ejecutar en cualquier otro ordenador Z80 de similares características que tuviera el mismo sistema operativo. El más conocido de tales sistemas en disco es el CP/M (*Control Program/Microcomputers*: programa de control para microodenadores).

Los sistemas operativos en disco como el CP/M son esencialmente un desarrollo de los monitores y sistemas operativos más específicos para una sola máquina, pero representan un importantísimo avance en términos de compatibilidad de software. Cualquier programa escrito para funcionar con un sistema operativo genérico como el CP/M o el MS-DOS funcionará en cualquier otro ordenador con ese sistema operativo, siempre y cuando el software no intente hacer uso de ninguna configuración especial (como, p. ej., efectos sonoros) específicos de una máquina. El software del sistema operativo llega a manos del fabricante de ordenadores en forma estándar, suministrado directamente por quienes lo desarrollan. Todo lo que el fabricante de hardware ha de hacer es reescribir una pequeña porción, dependiente de la máquina, del programa.



Ian McKinnell

La clave del éxito
El Osborne 1 debe gran parte de su éxito al hecho de ser un ordenador basado en CP/M. Viene con algunos de los programas CP/M más populares, incluyendo el WordStar, SuperCalc y MBASIC

Los sistemas operativos en disco varían considerablemente en cuanto a complejidad y capacidades, pero los más simples, como el CP/M y el MS-DOS, constan esencialmente de tres partes: el procesador de órdenes, el sistema operativo en disco básico (*Basic Disk Operating System*: BDOS) y el sistema básico de entrada-salida (*Basic Input/Output System*: BIOS). De estos componentes, el único que tiene relación con la compatibilidad es el BIOS. Éste es una parte separada del programa que contiene todas las rutinas necesarias para manipular los periféricos, incluyendo la pantalla y el teclado, y su configuración debe ser especial para cada nuevo diseño de ordenador. Todo programa que se ejecute bajo el control de este sistema operativo se conectará en interface con el ordenador a través del BIOS. Éste, entonces, asumirá funciones como recoger los caracteres del teclado, dar salida a los caracteres hacia la pantalla o la impresora, direccionar los discos, etc.

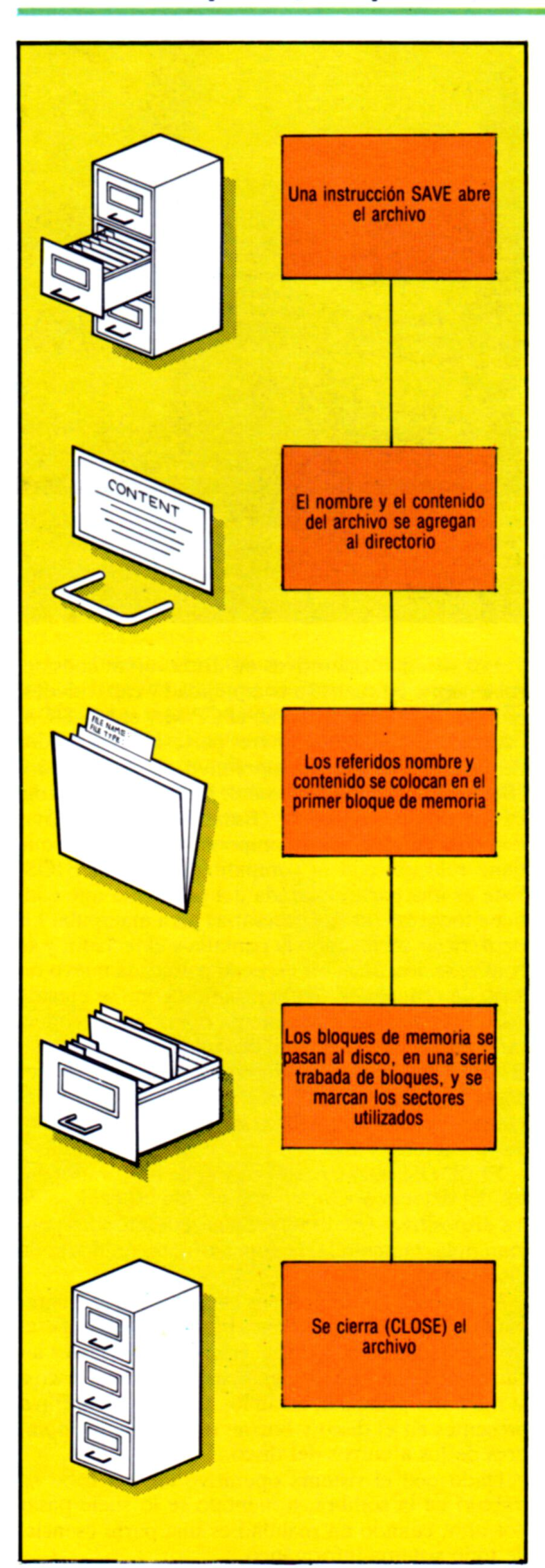
El BDOS, por su parte, consta de los componentes del sistema operativo que no son específicos de los dispositivos (es decir, rutinas generalizadas para manipular la pantalla, la impresora, las unidades de disco, etc.).

El procesador de órdenes es la parte del programa que manipula las órdenes para el sistema operativo digitadas en el teclado. Las órdenes usuales incluyen aquellas que cargan archivos desde el disco a la memoria principal, listan los nombres de archivo presentes en el disco y borran o cambian los nombres de los archivos del disco.

Dado que el sistema operativo lleva a cabo su trabajo en la sombra, a menudo se lo suele pasar por alto, cuando en realidad es una parte esencial de todo sistema informático.

Amos de la memoria

Iniciamos una serie dedicada al estudio de los archivos: qué son, cómo se emplean, de qué métodos se dispone para acceder a ellos



Cómo se guarda un archivo binario
Un archivo binario es simplemente la copia de un trozo de la memoria. Puede contener tanto un programa que encierra la memoria, como una imagen que aparece en pantalla, como cualquier otra cosa. El proceso archivador es inmediato: tras ser apuntada su entrada en el directorio, los datos de archivo se escriben como una cadena de sectores trabados entre sí. El DOS también llevará una lista de estos sectores, de modo que cuando haya que incorporar otro archivo no se escriba sobre los utilizados

Archivo es la palabra justa cuando se habla del almacenamiento informático, puesto que se pueden establecer analogías directas con el método de almacenar documentos e informes en un sistema de archivo tradicional. Teniendo esto presente, veamos primero por qué son necesarios los sistemas de archivo en los ordenadores personales.

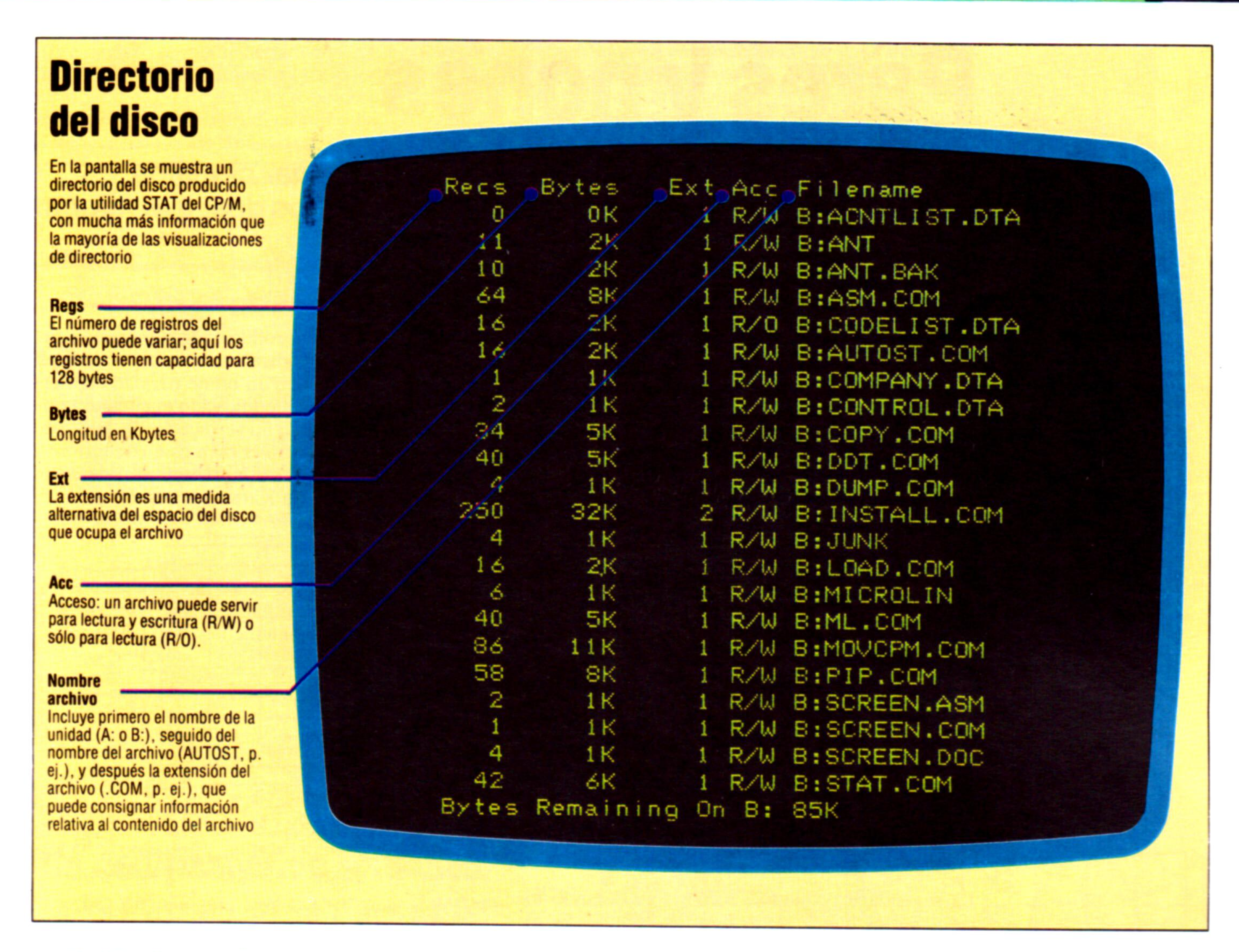
Con el fin de "administrar" eficazmente nuestra vida cotidiana, es necesario mantener el registro más exacto posible de nuestras experiencias, transacciones monetarias, citas sociales, etc. La mayoría de las personas utiliza un diario-agenda y lleva el estado de una cuenta bancaria a través de los resguardos de un talonario. Esta necesidad de almacenar información y, lo más importante, de recuperarla fácil y simplemente, se amplía muchas veces cuando hay que tratar con una gran cantidad de información constantemente cambiante, típica de cualquier empresa o proyecto que abarque distintas personas, lugares, objetos y circunstancias. A medida que estas empresas van creciendo, el manejo de la información se va haciendo cada vez más complejo, y muchos de los problemas con que se encuentran las empresas de este tipo surgen por causa de su mala administración y una interpretación errónea de la información. Un sencillo método para almacenar, llevar un índice y recuperar los datos puede ser la clave de una buena administración. Los directores valoran la necesidad de buenas técnicas de archivo, y ajustan los métodos de almacenamiento de acuerdo con el tipo, el volumen y el porcentaje de variación de la información que tienen bajo su control.

Estos principios son válidos cuando los aplicamos a los sistemas de que dispone un ordenador para manipular y almacenar con precisión enormes cantidades de información a una velocidad increíblemente elevada. En el centro de un sistema de este tipo se halla el "administrador" del ordenador: el sistema operativo en disco (DOS). Éste trabaja de manera perfecta siempre que su "jefe" (es decir, el usuario o su programa) le proporcione la información correcta y le formule las preguntas adecuadas. Por tanto, los sistemas de almacenamiento por ordenador son eficaces si también lo son la organización de los datos (o sistema de archivo) adoptada por el DOS y la forma como los utiliza el DOS.

Los métodos estándar de tratamiento de archivos que utilizan los microordenadores en un principio se pensaron para ordenadores de unidad principal y miniordenadores. Se dividen en tres clases: archivos binarios, archivos secuenciales y archivos de acceso directo. Vamos a analizarlos por separado.

Archivos binarios

Un archivo binario es, simplemente, la copia de una porción de la memoria reservada al usuario. Un buen ejemplo sería un programa guardado (SAVE). Imaginemos esa zona de la RAM como un



sencillo bloc de notas. Si a usted le interesara conservar las notas principales o los apuntes más interesantes, arrancaría las páginas correspondientes y las conservaría en algún sitio donde las tuviera a mano. Los archivos binarios funcionan de la misma manera. Cuando se imparte una orden SAVE, el DOS almacena el NOMBRE ARCHIVO en disco, marcándolo de alguna forma especial como un archivo binario, y después copia el área correspondiente de la memoria byte por byte en el disco. El programa se almacena en bloques unidos (con los señaladores al final de cada bloque indicando dónde comienza el bloque siguiente). El último bloque termina con un señalador de final de archivo. Utilizando nuestra analogía, podríamos decir que nosotros le hemos puesto un nombre a una página de nuestro bloc de notas y a continuación hemos procedido a almacenarla en un cajón del archivador, añadiendo después el nombre de esa página al índice de contenidos del cajón.

Cuando se empieza (RETURN) o se acaba (ENTER) una línea de programa, ésta se comprime en una forma distintivada: el intérprete de BASIC toma las palabras clave y las codifica mediante números de un byte. Éstos se pueden manejar más fácilmente y decodificar cuando se reclama un listado del programa (LIST). Como un archivo binario es una imagen de la RAM, también puede almacenar códigos ASCII y datos en binario. La facilidad de guardar archivos en ASCII es útil, por ejemplo, para almacenar el contenido de la memoria de pantalla, de modo que las visualizaciones en pantalla se puedan guardar (SAVE) para volverlas luego a cargar (LOAD) en la misma zona de la memoria. Además, algunos sistemas operativos en disco y algunas versiones de BASIC permiten almacenar un programa en forma ASCII. Ello posibilita editar el programa no distintivado como un archivo de texto en las máquinas con programas de muy elaborada edición.

Los archivos binarios son muy sencillos de utilizar y administrar, pero están limitados por dos factores. El primero de ellos es que sólo se puede guardar la información correspondiente como un todo. Por consiguiente, la información se debe recuperar de la misma manera y, en consecuencia, los archivos binarios se deben cargar otra vez en la memoria en su totalidad. En segundo lugar, las dimensiones máximas de un archivo quedan en todo caso limitadas por la cantidad de RAM disponible para el usuario.

En el próximo capítulo de este curso analizaremos los archivos secuenciales, que permiten que un archivo sea tan largo como se necesite (dentro de los límites del espacio del disco) y los archivos de acceso directo, que emplean métodos diferentes para permitir que el DOS almacene los datos de tal forma que luego se puedan recuperar y actualizar libremente.

Pares y nones

Nuestro curso de lógica nos permite ahora diseñar circuitos bastante complejos. Ahora vamos a analizar otros dos, que figuran entre los más importantes de un ordenador

Antes de empezar a analizar el diseño de estas dos aplicaciones avanzadas, primero analizaremos con todo detalle otra puerta lógica importante: la puerta OR-exclusivo (XOR). Esta puerta ya la consideramos someramente (véase p. 527), pero todavía no habíamos ofrecido ni su símbolo de álgebra booleana ni el de su diagrama de circuitos:

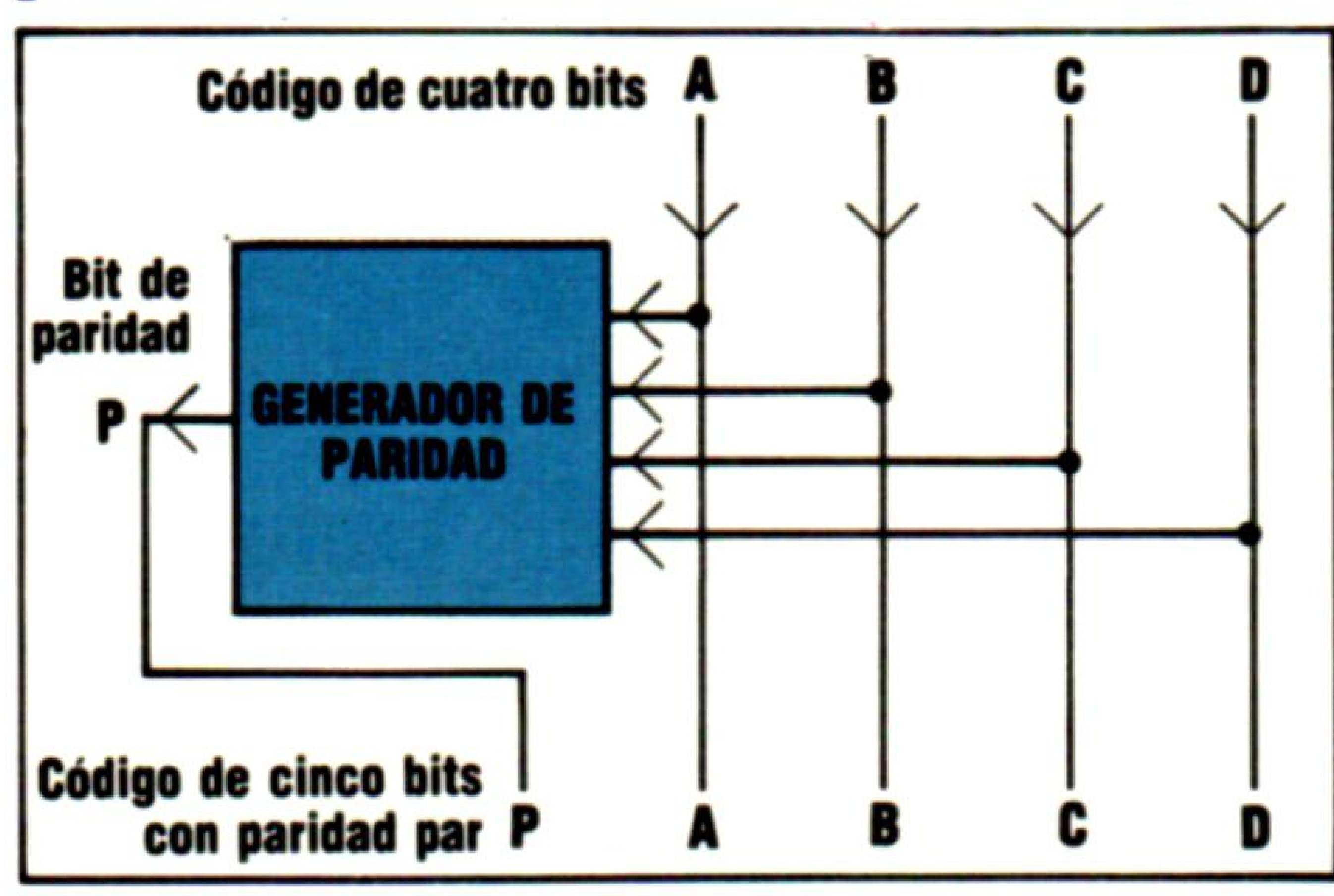
Tabla de verdad			Símbolo de circuito
A	B	C	
0	0	0	A — XOR — C
0	1	1	B
1	0	1	Símbolo booleano = ⊕
1	1	0	

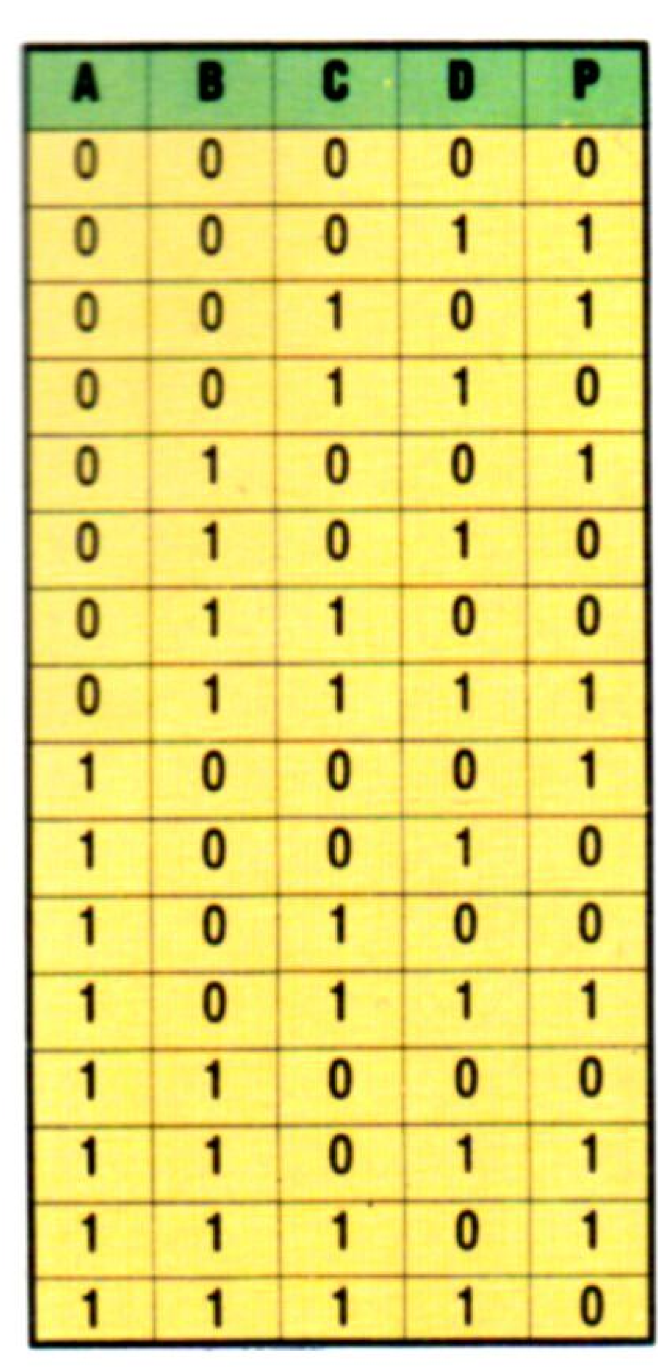
A partir de la tabla de verdad se puede ver que la salida C se puede expresar de dos formas:

a) $C = A \oplus B = \overline{A}.B + A.\overline{B}$
b) $\overline{C} = \overline{A \oplus B} = \overline{A}.\overline{B} + A.B$

La segunda expresión se forma considerando los casos en los que C no es uno (es decir, es cero). Esta puerta puede ser de particular utilidad para nuestra primera aplicación.

Un generador de bits de paridad

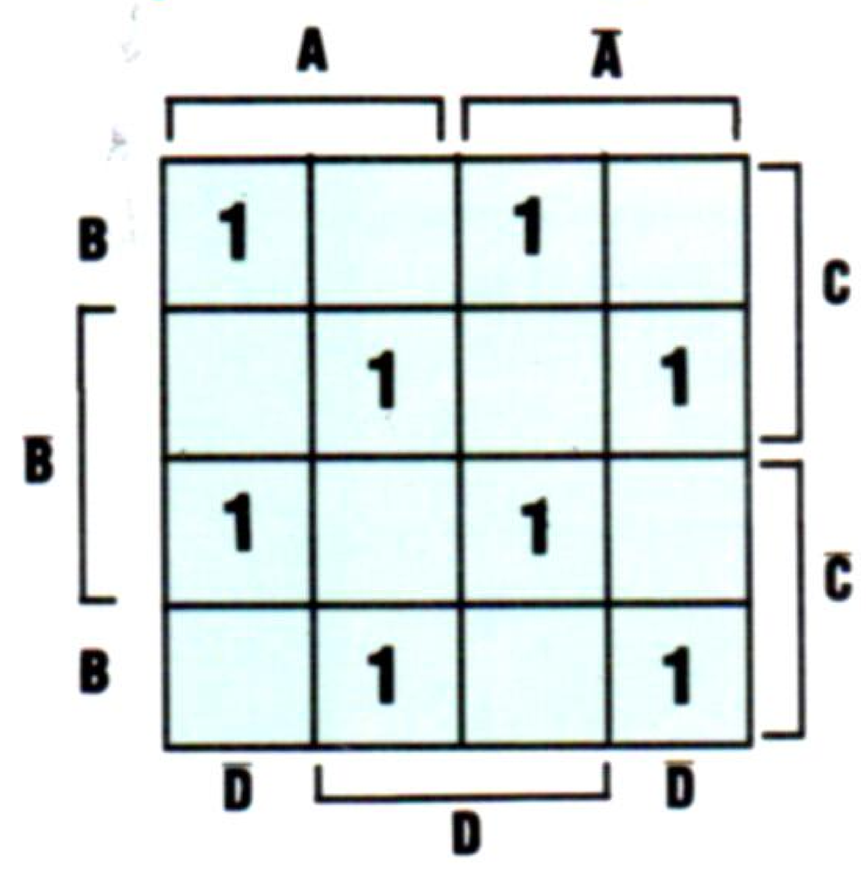




A	B	C	D	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Tabla de verdad de un GBP

El de paridad es un concepto importante en el diseño de sistemas de transmisión de datos. El bit de paridad (véase el diagrama) se agrega a los restantes bits de un código en binario con el fin de que todos los códigos binarios transmitidos posean un número par de unos. (Otra convención podría ser que todos los códigos contengan un número impar de unos, lo que se conoce por *paridad impar*.) Un bit de paridad actúa como un sistema de control para verificar si se ha producido la transmisión correcta. El circuito que diseñaremos acepta códigos de cuatro bits y produce un bit de paridad adecuado. Con una pequeña modificación el circuito también podría actuar como un verificador de datos entrantes. La tabla de verdad para este circuito está consignada al margen. La representación de estos valores en diagrama de Karnaugh nos da:



Esta figura del diagrama K por desgracia no es simplificable a la manera habitual, porque no se ha formado ningún grupo. La expresión para P es:

$$P=\overline{A}.\overline{B}.\overline{C}.D+\overline{A}.\overline{B}.C.\overline{D}+\overline{A}.B.\overline{C}.\overline{D}+\overline{A}.B.C.D$$
$$+A.\overline{B}.\overline{C}.\overline{D}+A.\overline{B}.C.D+A.B.\overline{C}.D+A.B.C.\overline{D}$$

Si ahora procedemos a agrupar entre sí los términos rojos y los términos azules, nos es posible simplificar la expresión:

$$P=(\overline{A}.\overline{B}+A.B).(\overline{C}.D+C.\overline{D})$$
$$+(A.\overline{B}+\overline{A}.B).(\overline{C}.\overline{D}+C.D)$$

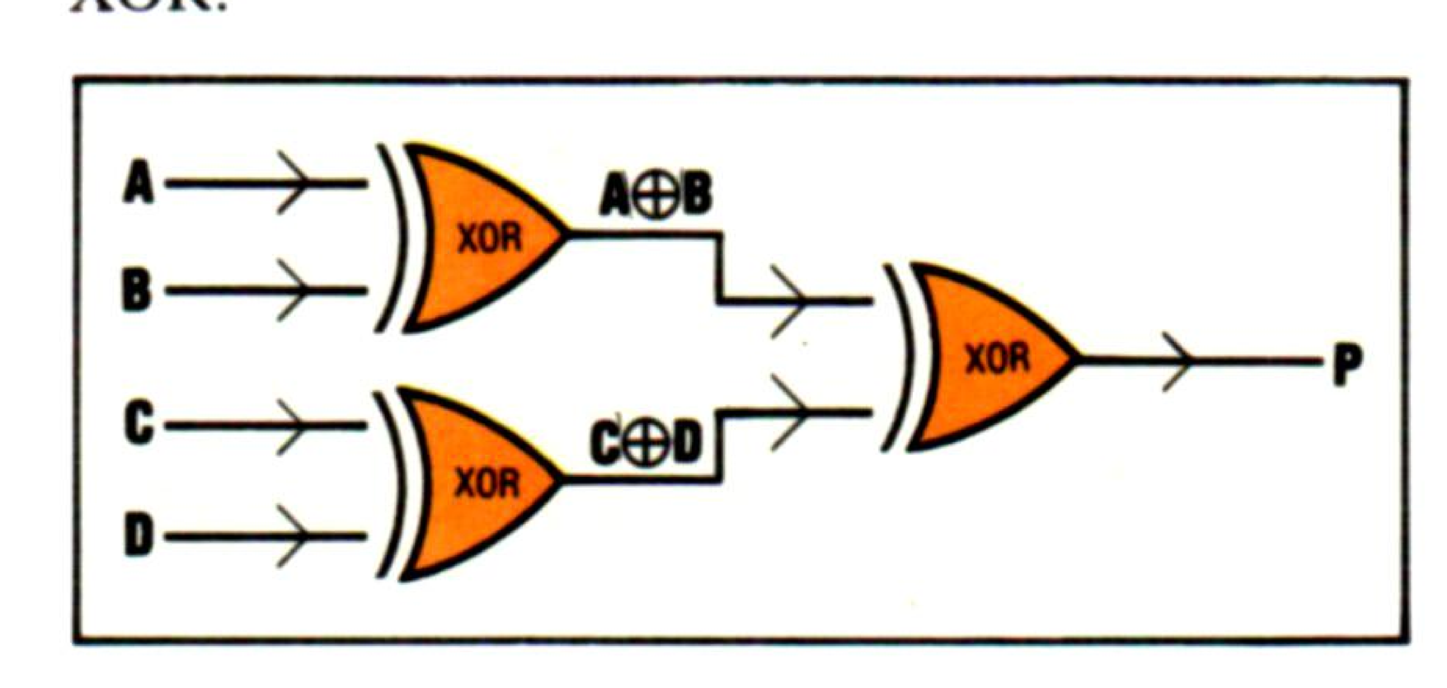
Ahora, remitiéndonos a las expresiones a) y b) para una puerta XOR que presentamos al comienzo de este artículo, podemos simplificar más la expresión para obtener:

$$P = (\overline{A \oplus B}).(C \oplus D) + (A \oplus B).(\overline{C \oplus D})$$

Considerando cada término entre paréntesis como una entrada para una puerta XOR, la expresión se podría reducir aún más:

$$P = (A \oplus B) + (C \oplus D)$$

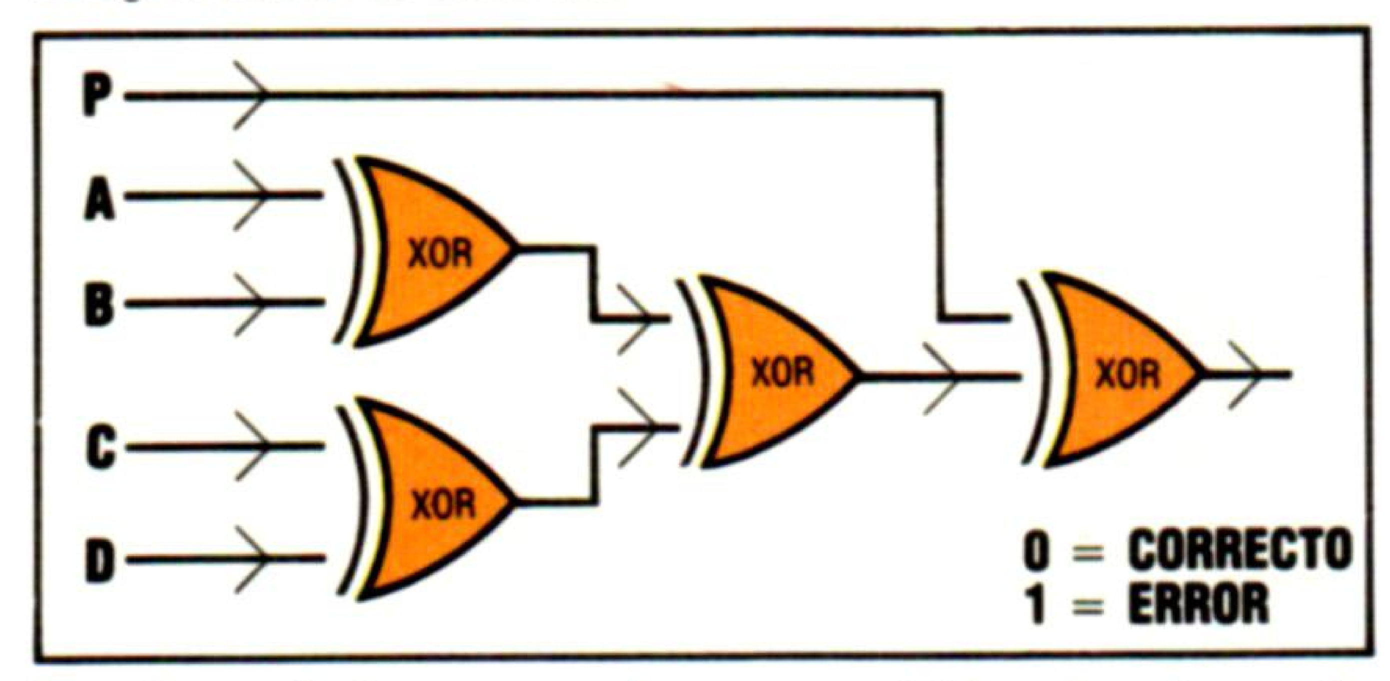
y formar el circuito como una cascada de puertas XOR:





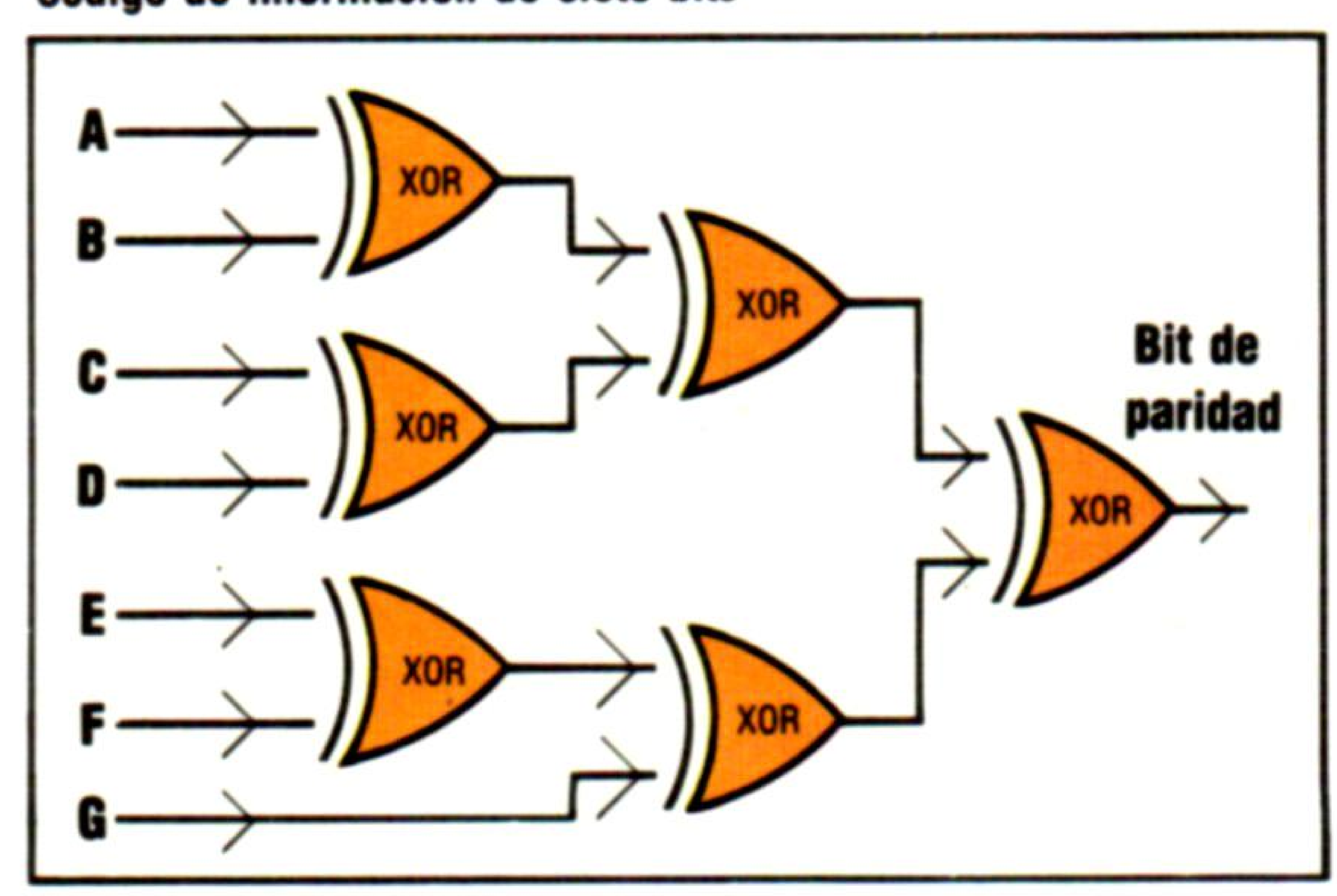
Este circuito se modifica para que actúe como control de paridad, añadiendo otra puerta XOR para comparar el bit de paridad recibido con uno generado por el circuito en el extremo de recepción.

Código recibido de cinco bits



En la práctica, para la transmisión de datos la mayoría de los ordenadores utilizan el código ASCII, que tiene ocho bits, siete de información y uno de paridad par. Es fácil concebir un generador de bits de paridad para códigos ASCII:

Código de información de siete bits



Un codificador de prioridad

Muchos ordenadores emplean sistemas "interruptores de prioridad" para controlar el flujo de datos hacia y desde dispositivos periféricos. En estos sistemas la operación de la CPU se interrumpe en virtud de una señal proveniente del periférico cuando éste necesita la atención de la CPU. Sin embargo, cuando dos o más periféricos interrumpen a la CPU al mismo tiempo, ésta ha de seguir un orden de prioridad para "atender" primero al dispositivo más importante. El codificador de prioridad que nosotros diseñaremos englobará cuatro dispositivos periféricos en un circuito que pueda identificar cuál es el periférico que está produciendo señales.

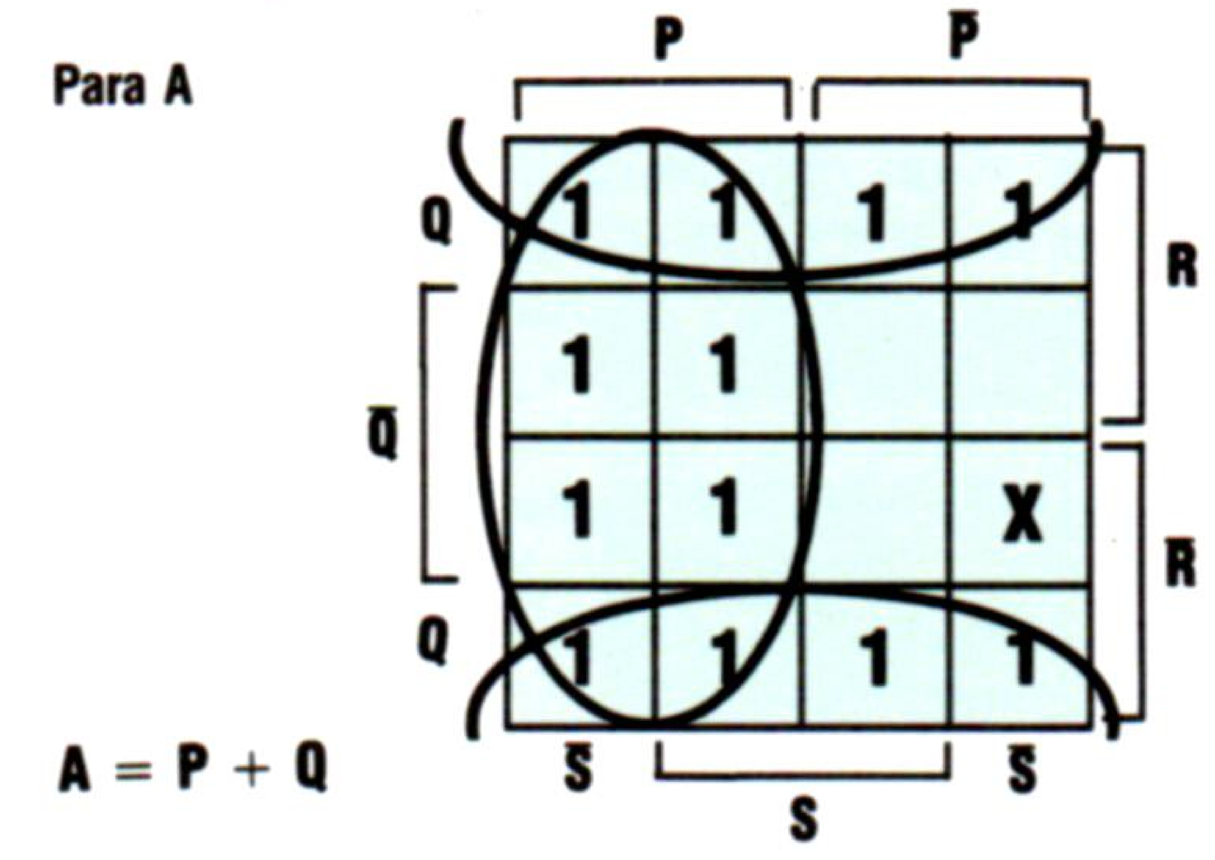
Para obtener información en uno de los cuatro dispositivos se necesitan dos líneas de salida, más una tercera línea de salida para indicar que se requiere una interrupción. Supongamos que los cuatro periféricos son P, Q, R y S, siendo P el de mayor prioridad y S el de menor prioridad. Las líneas de salida se denominarán A y B, para identificar al periférico, y Z para señalar una interrupción. La tabla de verdad para el codificador se puede realizar utilizando una X para aquellas situaciones ante las cuales el codificador "se muestra indiferente".

P	Q	R	S	A	B	Z
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Para ayudarle a comprender cómo se ha hecho esta tabla, observe la última fila. Aquí P está indicando una interrupción, y como P es el dispositivo de máxima prioridad, a nosotros no nos preocupa que los dispositivos de inferior prioridad señalen su prioridad o no.

Las tres salidas del circuito, A, B y Z, se deben analizar de forma independiente. Empezando por A, el diagrama K es:

Para A



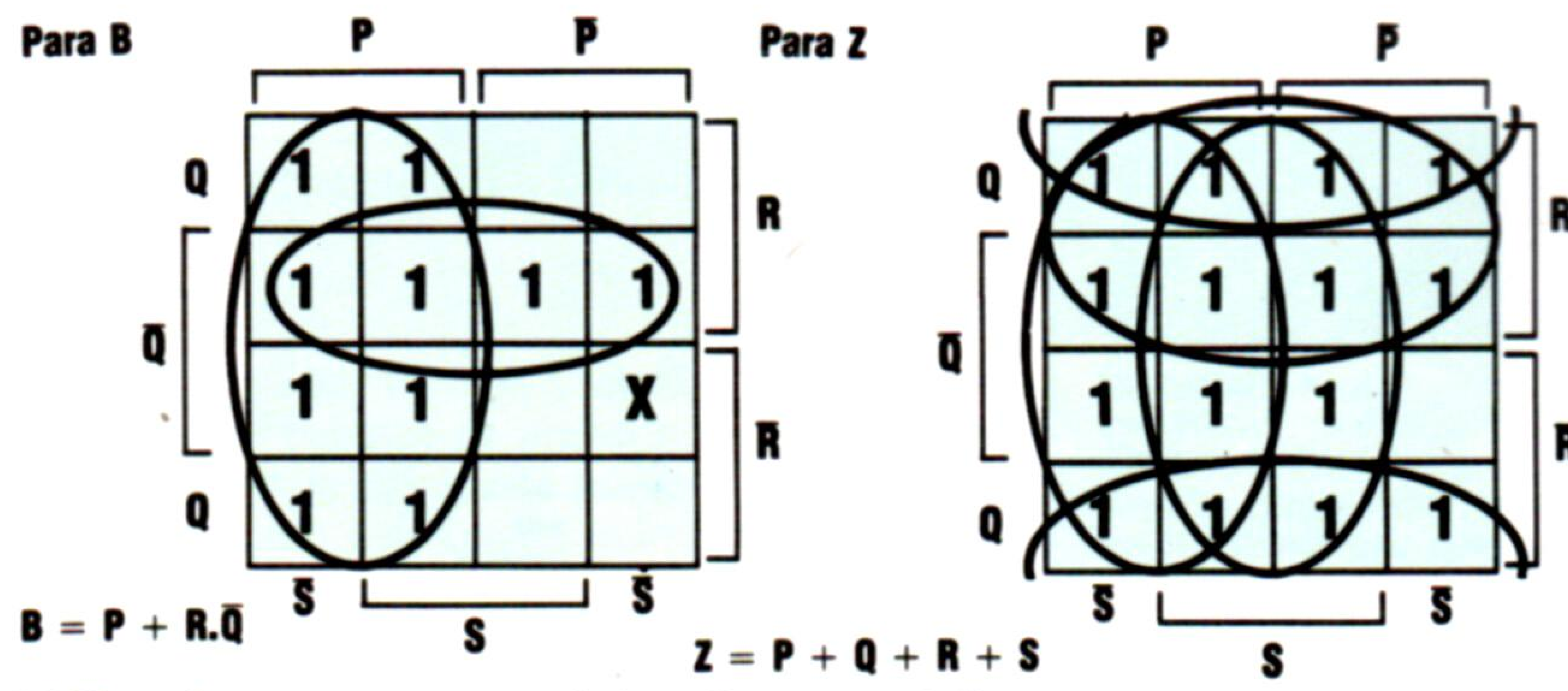
A = P + Q

El caso "indiferente" en la salida A está representado mediante una X (pero los casos "indiferentes" en las entradas se tratan de distinta forma). Tomemos el caso en el que P es uno y Q, R y S son "indiferentes". Aquí debemos rellenar todos los casilleros del diagrama K donde P sea uno; hay ocho en total. Del mismo diagrama sacamos la expresión simplificada:

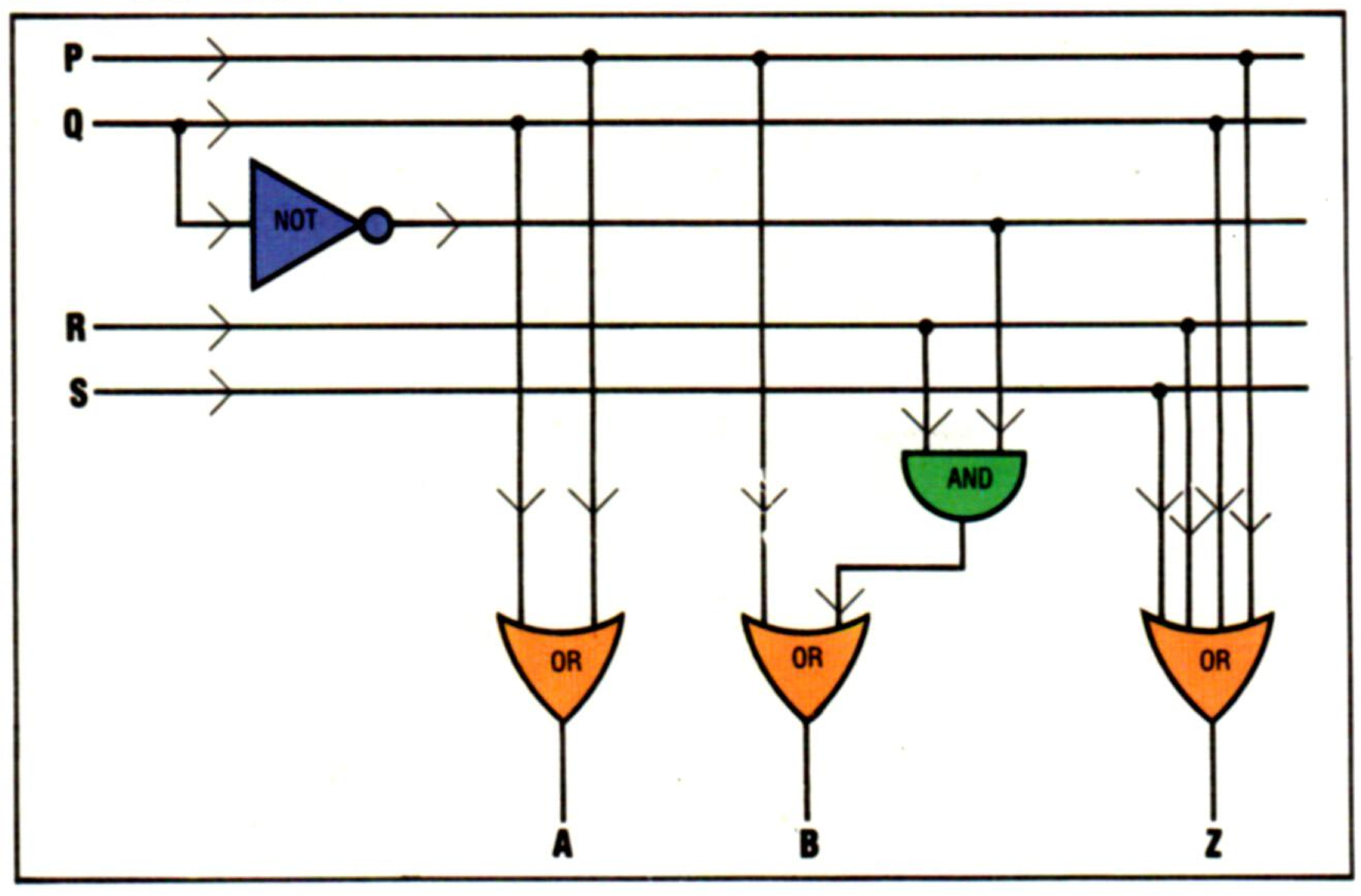
A = P + Q

Del mismo modo, para B y Z los diagramas K son:

Para B **Para Z**



Utilizando estas tres expresiones, llegamos al diseño de este circuito:



Contar los pasos

Los contadores son registros que desempeñan un papel fundamental en las reiteraciones

Un contador es un registro que sirve para indicar las veces que se ha repetido el ciclo de operaciones de un bucle. Veamos su importancia:

La figura 1 muestra cómo tras la introducción de un número por teclado se visualiza el total resultante de sumarlo con los que se introdujeron antes que él. En este ejemplo, ningún registro nos indicará cuántos números se desea entrar. De aquí que se haga una pregunta tan vaga como la de que si quedan más números, para poder salir del bucle. Este planteamiento es válido para casos en que se desconoce el número exacto de cantidades a las que se va a dar entrada. Pero si ya se conoce de antemano de cuántas cifras se va a disponer, puede que resulte más cómodo contar cuántas veces habremos de repetir un mismo ciclo de operaciones, utilizando para ello un contador.

La figura 2 representa el mismo ejemplo anterior pero sabiendo que hay un total de 10 números a sumar. Se observa la utilización de un registro llamado contador, que se pone a 1 en un momento determinado del programa y, tras la operación de sumar el total actual más el número recién introducido por teclado, se incrementa en una unidad. Acto seguido se controla su valor. Caso de que el contador no haya rebasado el número 10 (cantidad de números que deben entrarse), el flujo retorna al principio del bucle. Sin embargo, al llegar al valor 11 se cumple la condición de que el contador es mayor de 10 y se baja en secuencia, abandonándose el bucle a la décima reiteración. Se ha conseguido detener un ciclo reiterativo mediante el control de las veces que debía repetirse, número de veces que coincide con el valor límite contenido en el contador.

El valor de un contador no necesariamente debe ser objeto de incremento, ya que también se puede partir de un valor inicial superior e ir decreciendo, también de uno en uno, con lo cual el valor final que deberá controlarse será 0, tal y como muestra la figura 3. También es claro que a un contador lo podemos hacer crecer o bien decrecer no solamente a "golpes" de unos. Con la expresión CONT = CONT + 3, por ejemplo, haríamos que el contador aumentara de tres en tres. Así, podría definirse el contador como un registro cuyo valor aumenta o decrece en cantidades constantes, de uno en uno, de dos en dos, de diez en diez, etc.

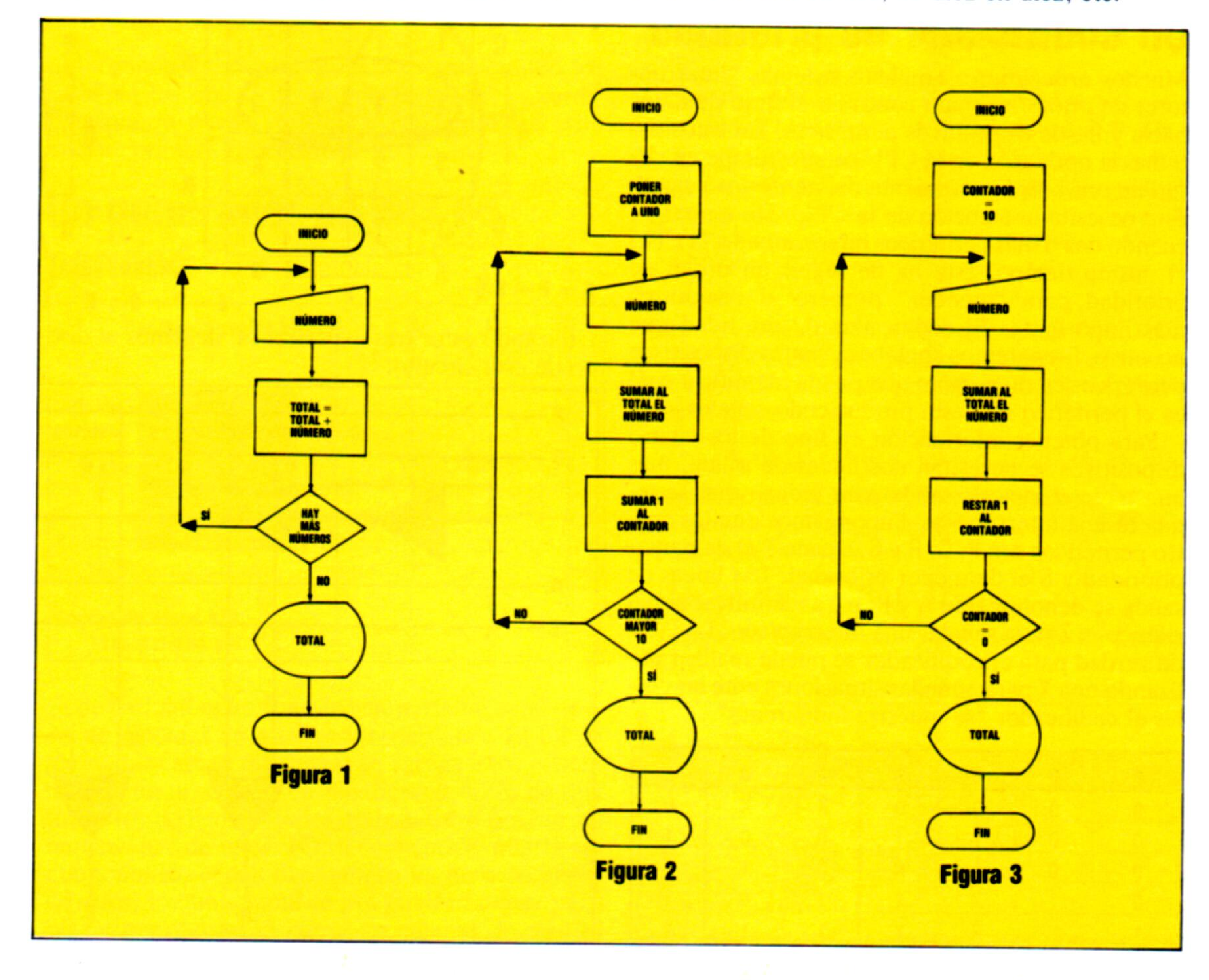


Figura 1 — Figura 2 — Figura 3



Atari se pone al día

Atari ha actualizado sus modelos 400 y 800, lanzando al mercado los ordenadores XL, más baratos y con atractivas configuraciones

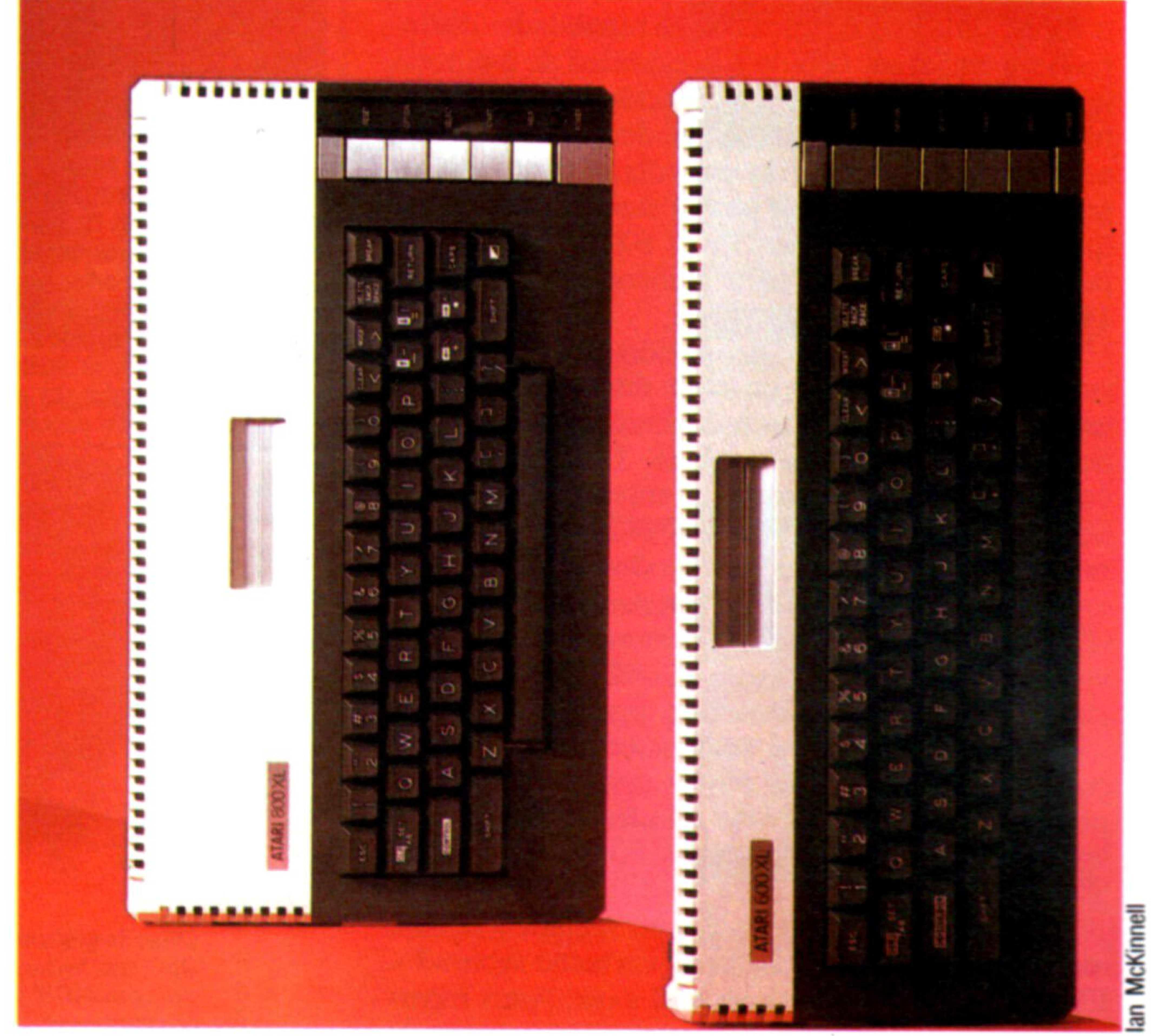
Enfrentada a la creciente competencia existente entre los fabricantes de ordenadores personales, Atari volvió a diseñar su línea de ordenadores y presentó el 600XL y el 800XL. Las nuevas máquinas pueden utilizar el software escrito para el 400 y el 800, que comprende una amplia gama de programas, desde los juegos más vendidos hasta paquetes de gestión.

Afortunadamente, ambas máquinas han adoptado el teclado del 800. El Atari 400 posee un teclado de tipo membrana, similar al del ZX81, con sus caracteres impresos sobre una membrana plástica plana. Debido al pequeño tamaño de las teclas y al grado de presión necesario para hacer contacto, la escritura al tacto es virtualmente imposible. Por el contrario, el 800 posee un teclado de tamaño natural, similar al de una máquina de escribir, que es uno de los mejores del mercado.

Los teclados de las nuevas máquinas son idénticos y poseen un total de 62 teclas. Éstas incluyen cuatro teclas de función: START, SELECT, OPTION y RESET, situadas a la derecha del teclado. También hay una tecla HELP, que trabaja con un software nuevo y proporciona útiles sugerencias en la pantalla. Hay 29 teclas para gráficos, y los de Atari poseen un juego de caracteres ASCII completo incorporado. Una original característica, heredada del 800, la constituye la manera como se utilizan las teclas del cursor. Las flechas están impresas como el tercer carácter de cuatro teclas separadas. Para desplazar el cursor se debe mantener deprimida la tecla CONTROL mientras se pulsa la tecla de flecha adecuada. Aparte de esta extraña configuración, los teclados están bien diseñados y es fácil digitar en ellos.

El Atari 400 viene con 16 Kbytes de memoria para el usuario, mientras que el 800 cuenta con 48 Kbytes. Ambos ordenadores poseen ranuras para ampliación en el tablero del sistema a las que se puede acceder quitando la tapa de la máquina. Los modelos XL no se deberían abrir. En cambio, las puertas para ampliación están incorporadas en la parte exterior de la carcasa. La única diferencia significativa entre ellos es la cantidad de memoria que viene como estándar. El 600XL viene con 16 Kbytes y se puede ampliar a 64 Kbytes enchufando un paquete de ampliación. El 800XL viene con 64 Kbytes.

Las dos máquinas XL poseen una interface incorporada que se denomina *Expander*. Ésta es una puerta para ampliación de tipo bus que se puede conectar a diversos periféricos y paquetes de ampliación. Asimismo, hay una ranura para cartuchos de ROM situada detrás del teclado, para juegos en cartucho y otro tipo de software. El Atari 800 original posee dos ranuras para cartuchos, pero virtualmente no se ha escrito software para hacer uso de la segunda ranura, de modo que en la línea XL ésta se ha suprimido. El 400 y el 800 poseen cuatro puertas para palanca de mandos, pero el software disponible no justifica este número, de modo que se han reducido a dos en las máquinas XL y se han desplazado desde delante de la carcasa hasta uno de los lados.



Ian McKinnell

Los gemelos Atari
Los ordenadores personales 600XL y 800XL de Atari son muy similares, pero el 600XL cuenta con 16 K de memoria, mientras que el 800XL dispone de 64 K. Las máquinas poseen un teclado de gran calidad y buenos gráficos en color. Puesto que se trata de versiones mejoradas de los Atari originales, disfrutan de una amplia gama de software

Luz y sonido

Todos los ordenadores Atari se pueden conectar directamente a un aparato de televisión y, además, excepto el 400, se pueden conectar a un monitor para ordenador. Conviene recalcar que la visualización en pantalla de Atari es de calidad, incluso en un televisor. El juego de caracteres generalmente es fácil de leer y el contraste entre texto y fondo es bastante bueno. Hay varias opciones de color disponibles, pero la visualización de textos normal se compone de letras blancas sobre un fondo de color azul oscuro. Los ordenadores Atari visualizan un máximo de 40 columnas por 24 líneas de texto. Existen cuatro modalidades para texto con visualizaciones distintas.

Los Atari fueron de los primeros ordenadores que proporcionaron gráficos sprite. La función sprite de los mismos se denomina gráficos *player-missile* y la controla un chip especial llamado chip GTIA. Los "jugadores" son objetos creados mediante pixels. Una vez se ha determinado la forma



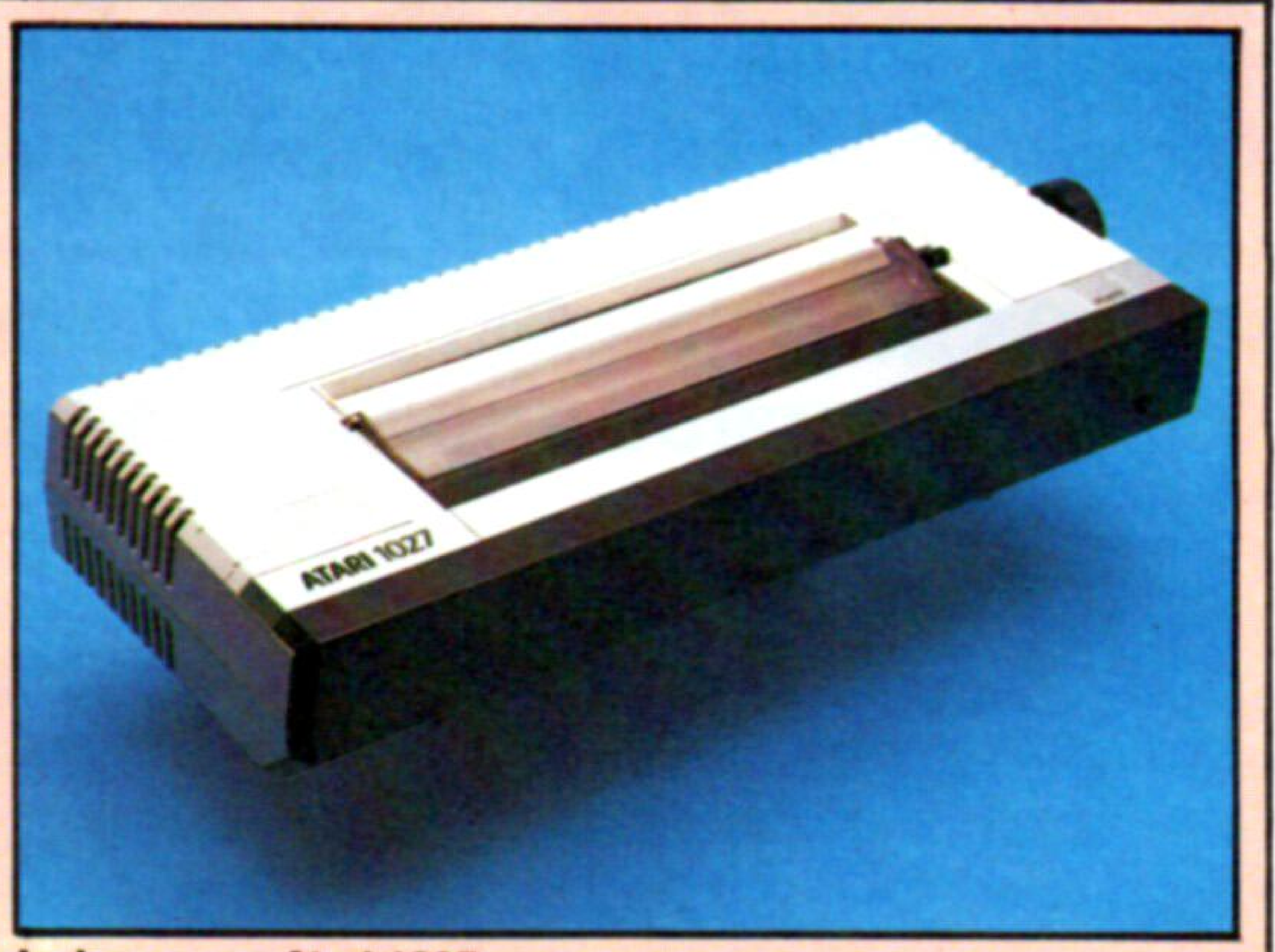
Unidad de disco
Una útil opción para el 800XL es la unidad de disco. Ésta proporciona 127 K de almacenamiento rápido. El 600XL estándar no posee memoria suficiente para utilizar la unidad de disco, pero se puede ampliar. Varios excelentes programas de juegos y de gestión se encuentran a la venta sólo en el formato de disco

de un jugador, los valores de cada pixel se colocan (POKE) en una zona de la memoria denominada *tabla de formas*. Uno puede crear hasta cuatro jugadores, cada uno de ellos con su correspondiente misil. Al jugador se le asignan uno o varios colores, y se lo manipula en la pantalla cambiando los colores de la tabla de formas. Aunque los gráficos *player-missile* de Atari no son fáciles de utilizar, facilitan notables visualizaciones en pantalla.

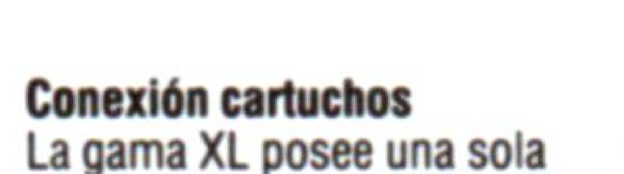
Las nuevas máquinas Atari poseen 11 modalidades para gráficos y hasta 256 colores (en realidad 16 colores, cada uno de ellos con 16 tonalidades distintas); sin embargo, debido a la cantidad de memoria que se requiere para la visualización en pantalla, el número de colores que se pueden mostrar varía según la resolución de la pantalla. Cuanto mayor sea la resolución, menor será la cantidad de colores que se puedan visualizar. En el 600XL y el 800XL la máxima resolución para gráficos es de 320 por 192 pixels.

Las configuraciones de sonido Atari también están controladas por un chip diseñado especialmente. Hay cuatro voces independientes, cada una de las cuales posee una escala de 3 ½ octavas. Las voces se pueden controlar a través de la orden SOUND en BASIC, o colocando (POKE) valores en los registros de la memoria que producen los diversos tonos. Los tonos también se pueden ajustar por oscilación, altura, distorsión y volumen. Cuando las voces se controlan mediante SOUND, sólo se puede emplear una voz cada vez. Ello significa que la armonización exige que se encienda cada voz por separado y la demora que hay es considerable. Este problema se puede superar empleando rutinas en lenguaje máquina o utilizando POKE en lugar de SOUND.

Los Atari 400 y 800 no tienen ningún lenguaje incorporado; se debe utilizar un cartucho separado. El 600XL y el 800XL poseen el BASIC Atari incorporado. Este BASIC no es el mejor, ya que carece de muchas de las configuraciones que hacen que el BASIC BBC y otros sean tan buenos. Por ejemplo, no tiene orden CIRCLE, ni la configuración PRINT@ ni PRINT USING, ni numeración o renumeración automática de líneas, ni provisión para variables de enteros. Sin embargo, se encuentran a la venta en cartuchos de ROM el BASIC Microsoft y el Extended BASIC.



La impresora Atari 1027
Posee una cabeza de impresión similar a la bola que usan algunas máquinas de escribir. Da una impresión tan buena como la de las máquinas eléctricas, pero es de operación más lenta. Existen otras dos impresoras Atari, una de las cuales usa lápices esferográficos para dibujar letras y líneas en cuatro colores; la otra es una impresora matricial que ofrece una impresión de inferior calidad, pero es rápida. Éstas son las únicas impresoras directamente utilizables con las máquinas XL, debido a la falta de una interface estándar para impresora



Conexión cartuchos
La gama XL posee una sola ranura para cartuchos de ROM



Chips de gráficos
Dos chips hechos a medida, llamados ANTIC y GTIA, proporcionan las espectaculares capacidades para gráficos de Atari



RAM
Numerosos chips componen los 16 K de RAM





Bola y palanca
Atari ofrece dos formas principales para controlar los juegos. El método convencional es mediante una palanca de mando, y la de Atari (a la derecha) se ha convertido en algo así como un estándar industrial. El mando de bola Atari (a la izquierda) controla a través de una bola que rota en su soporte

Para acompañar a la nueva línea de ordenadores, Atari ha vuelto a diseñar los periféricos existentes y ha incrementado el número de opciones disponibles para ampliación. Quizá el periférico más útil sea una caja de ampliación, que se enchufa en la interface *Expander*. La caja de ampliación proporciona ocho ranuras para ampliación, que pueden albergar fichas de interface para varios periféricos, dos puertas en serie RS232 y un bus en paralelo. Atari también posee un módulo CP/M con un microprocesador Z80, el sistema operativo CP/M 2.2 y una visualización conmutable de 40/80 columnas.

Con el 600XL y el 800XL Atari ha alcanzado cotas muy elevadas en cuanto a diseño, calidad de construcción y configuraciones. Ambos disponen de una vasta biblioteca de software en cartucho, cassette y disco que ha ido creciendo en los últimos años. Estas dos máquinas de Atari supondrán una adquisición muy atractiva para los entusiastas de los ordenadores personales.



ROM
Dos chips de ROM retienen el intérprete de BASIC

Conexiones periféricos
Se utiliza una puerta de 13 patillas para conectar los periféricos, incluyendo unidades de disco, impresoras y la grabadora de cassette exclusiva

Conexiones palancas de mando

Chip de sonido
Un chip construido a la medida y denominado POKEY se encarga de la generación de sonido

Chip de E/S
Un 6520 manipula las puertas de entrada y salida

CPU
El Atari se basa en un veloz chip 6502

La grabadora de cassette Atari
Los Atari sólo funcionan con su propia grabadora de cassette. Si bien ello encarece el precio del sistema, tiene sus ventajas. En primer lugar, al ser de Atari, la grabadora es más fiable. En segundo lugar, utiliza dos canales. Uno es para guardar programas en la forma normal; en el otro se puede grabar sonido. Esto permite que los programas para aprender idiomas reproduzcan fragmentos hablados en el momento preciso

ATARI 600XL/ 800XL

DIMENSIONES
600XL: 380 × 170 × 40 mm
800XL: 380 × 220 × 40 mm

CPU
6502, 2 MHz

MEMORIA
16-64 Kbytes de RAM, 24 K de ROM

PANTALLA
Hasta 24 filas de 40 columnas de texto, gráficos de hasta 320 × 192 con sprites y 16 colores con 16 niveles de brillo

INTERFACES
Palancas de mando (2), puerta para periféricos, puerta para ampliación, puerta para cartuchos

LENGUAJES DISPONIBLES
BASIC, FORTH, LOGO, PILOT, lenguaje Assembly 6502

TECLADO
Estilo máquina de escribir con 62 teclas, incluyendo teclas de cursor y de función exclusiva como SELECT y START para control de programas

DOCUMENTACION
Los manuales nunca han sido punto fuerte de Atari, ya que la empresa ha tendido a considerar sus ordenadores básicamente como máquinas para juegos y se ha saltado los detalles técnicos. No obstante, hay una enorme gama de soberbios manuales y revistas independientes, aunque éstos suponen un gasto extra para los usuarios de Atari

VENTAJAS
Los ordenadores Atari continúan ofreciendo lo mejor en cuanto a juegos y una amplia gama de software. Las nuevas máquinas también proporcionan excelentes opciones para ampliación

DESVENTAJAS
Los Atari pueden ser caros: se necesita una grabadora de cassette exclusiva y el precio del software suele ser elevado. Asimismo, la programación de gráficos y sonido es más difícil que en otras máquinas

El control del stock

La codificación es la base de aquellos sistemas de gestión que tratan con toda clase de detalles el almacén. Estudiaremos ahora las exigencias de diseño en un paquete concreto

Hemos estudiado con algún detalle las distintas formas de clasificar las existencias en el archivo de datos de almacén. Además de emplear sencillamente un número de código, el sistema debe permitir que el usuario grabe información en cada tipo de existencias.

Esta información necesitará clasificarse por categorías de datos relativamente constantes y la cantidad de datos se determinará en función de la capacidad de almacenamiento de su ordenador. El diseño de un sistema de este tipo deberá, por consiguiente, economizar detalles.

El *Stock recording system* (sistema de control de existencias) de Dragon Data para el Dragon 64 posee siete *campos* en los que grabar información para cada registro de existencias. Se reservan para el número del artículo, su descripción, nuevos pedidos, el precio de costo, precio de venta y la unidad de medida.

Los campos forman una estructura importante en el control de almacén. La diferencia entre el precio de coste y el precio de venta nos dará el beneficio bruto. Gracias al número los artículos también se pueden agrupar, lo que ayuda a analizar y resumir los informes. El campo de unidad de medida es esencial porque las mercancías se embalan o envasan de diversas maneras.

Tal vez el aspecto más interesante del diseño de un sistema de control de almacén sea el hecho de que gran parte de la información requerida se compone de datos *dinámicos* en vez de *estáticos*. Ya hemos visto que los registros del libro mayor suelen contener información relativamente constante acerca del cliente o de la cuenta (que ocupa la parte de encabezamiento del registro) e información relativa a las transacciones efectuadas en dicha cuenta.

En el caso del control de existencias, sin embargo, las líneas divisorias que existen entre la parte estática del registro y la parte dinámica o basada en las transacciones del registro, se vuelven menos definidas. El nombre de la existencia y la descripción de grupo así como la codificación son el equivalente de la información estática de los registros de clientes o proveedores en los sistemas de libro de compras o ventas. Pero en un paquete para control de existencias, los datos estáticos se han de complementar con una gran cantidad de información derivada de las transacciones de existencias.

Por ejemplo, la dirección postal o el número de teléfono de los clientes en un archivo maestro del libro de ventas cambian con una frecuencia relativamente escasa. Por consiguiente, el programa tendrá una opción de mantenimiento del archivo maestro que permitirá que usted rectifique los registros de clientes particulares.

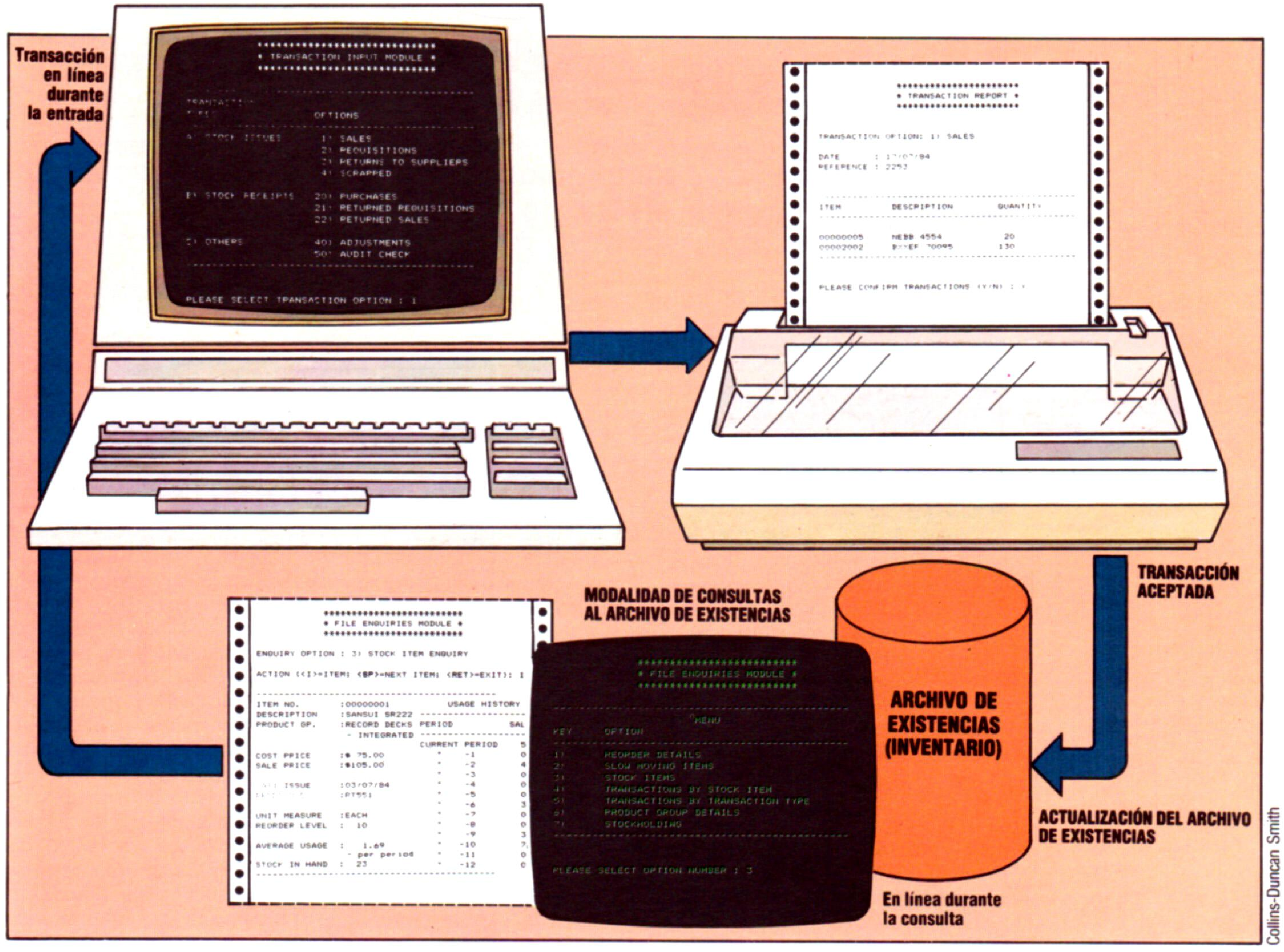
Pero si tomamos como ejemplo los campos de precio de venta y precio de costo del artículo del almacén, éstos pueden variar cada vez que la empresa vuelve a abastecerse de existencias. En este caso la solución lógica es que el programa tome esta información (junto con la fecha del cambio de precios y el volumen de existencias afectado por el mismo) desde una rutina de entrada de transacciones que registre los bienes recibidos, y no desde un programa para mantenimiento de archivos.

Para llegar a comprender cómo opera un programa para control de existencias es necesario que analicemos las rutinas de entrada y las facilidades para consultas e informes que las mismas ofrecen. En la ilustración gráfica hemos tomado como ejemplo el programa de Dragon señalando los diversos archivos. Los tres elementos importantes son: las rutinas de entrada de las operaciones de tráfico, detalles de las mismas y un archivo que informe sobre los artículos en existencia.



Un almacén de respuestas
El *Stock recording system* es uno de los muchos programas de aplicaciones para el Dragon 64 escritos para operar bajo el OS9. Se trata de un sistema operativo multiprogramación y multitarea que desarrolló Dragon a partir del sistema operativo UNIX

Ian McKinnell



Collins-Duncan Smith

En el sistema Dragon todas las visualizaciones de entrada de una operación tienen el mismo trazado. Los tipos de operaciones mercantiles se explican más o menos por sí solos. Por el momento nos ceñiremos a las ventas (es decir, a los movimientos hacia afuera del registro de existencias). Pero antes es interesante observar que todos los programas de gestión se diseñan para que resulten cómodos y fáciles de aplicar, y que le indicarán al usuario que dé entrada a toda la información necesaria. Esto nos lleva a dos exigencias de programación diferentes que se han de satisfacer para el logro perfecto de un sistema.

De una parte, el programa debe reconocer ciertos campos de datos y realizar sobre los datos ciertas manipulaciones, aritméticas, de clasificación, etc. Por otra parte, el programa ha de guiar al usuario indicándole los datos para los que no ha dado una entrada apropiada.

Una vez que el usuario ha dado entrada a los datos, al número de referencia para documentar la entrada (número que corresponderá al de la factura de la venta) y al número del artículo, el ordenador lee el archivo de existencias para ver si el artículo en cuestión existe. De hallar el número, visualiza automáticamente la descripción asignada a ese número de artículo. Tal visualización actúa como un verificador óptico por el cual el usuario entiende que puede dar entrada a la cantidad vendida.

A partir de esta información el ordenador está en condiciones de extraer una gran cantidad de detalles e informes. Por ejemplo, una de las opciones del menú principal, FILE ENQUIRIES (consultas al archivo), posee un submenú que consta de siete opciones que tratan los artículos en existencia, las transacciones y los artículos por grupos. He aquí las opciones: STOCK DETAILS (detalles de las existencias), SLOW MOVING ITEMS (artículos de movimiento lento), RE-ORDER DETAILS (detalles de nuevos pedidos), TRANSACTIONS (BY STOCK ITEMS) (transacciones por artículos de almacén), TRANSACTIONS (BY TRANSACTION TYPE) (operaciones según el tipo de transacción), PRODUCT GROUP DETAILS (detalles por grupos de productos) y STOCKHOLDING (conservación de existencias).

Las transacciones de venta afectan a todos estos informes. Si se solicita, por ejemplo, un informe sobre nuevos pedidos, el programa verificará si las existencias vendidas han hecho que la cantidad de las disponibles caiga por debajo del nivel especificado para volver a efectuar un pedido.

Todos los detalles introducidos en la visualización de una venta están correlacionados y se utilizan de una forma u otra. La visualización de consulta del programa Dragon proporciona una clara ilustración acerca de cuánta información de tipo administrativo se puede obtener a partir de una anotación de las ventas. En el tratamiento de históricos se ofrecen, inmediatamente calculados, el flujo y el volumen de la mercancía movida durante el año, el promedio de uso por período, y también se visualizan la fecha y referencia de la última operación.

Entrada documentada
El programa de control de existencias recibe información sobre una operación y comprueba la autoridad del usuario para dar entrada a información, así como que las transacciones estén correctamente anotadas. El archivo de existencias, que contiene los registros de artículos de existencias individuales, está en línea durante este proceso, y se actualiza. El programa también contiene un módulo de base de datos, que permite al usuario inspeccionar el archivo de control de existencias y crear informes relativos a diversos aspectos del inventario

Hora de examen

En esta ocasión nos detendremos para realizar un breve recuento del trabajo que hemos efectuado hasta ahora

El curso lo empezamos sugiriendo que para el trabajo resulta esencial disponer de las herramientas adecuadas (véase p. 524). Hay una gran variedad de equipos que resultan idóneos para las tareas de construcción, modificación o reparación. Usted *puede* utilizar un destornillador plano para un tornillo cruciforme, pero es bastante probable que si lo hace los estropee a ambos.

Cuando se trata de separar y unir cables ocurre lo mismo. Una buena juntura de soldadura durará más y funcionará mejor que una que se ha conectado torpemente. Aplique siempre la soldadura al cable y no al soldador, porque la soldadura irá corroyendo lentamente la punta de su soldador. Cuanto menos contacto haya entre ambos, mejor será. Deje que la soldadura fluya a lo largo de los dos cables que desea unir y sólo cuando la juntura esté cubierta totalmente quite el calor y enfríe la juntura soplando suavemente sobre ella. Siguiendo estas reglas se asegurará de que la juntura no esté seca. Desoldando usted puede quitar componentes de los tableros de circuitos con toda seguridad con el fin de repararlos o sustituirlos (véase p. 548).

Hemos analizado los conceptos fundamentales de la electrónica digital, tanto en el curso de *Bricolaje* como en el de *Ciencia informática*. Hemos presentado los componentes básicos y hemos visto cómo se interrelacionan. La resistencia es el más sencillo de todos los componentes, y el que se usa con más frecuencia. En el interior de su ordenador, verá más resistencias que circuitos integrados.

El condensador también es muy común. En los ordenadores, éstos se utilizan para filtrar el ruido que se acopla a una señal. Cada vez que una señal se modifica, se amplifica o se emplea de alguna otra forma, se degrada. De modo que es esencial algún medio de restaurar la señal a su forma incorrupta. El condensador es la forma más simple de eliminar los elementos que pudieran desvirtuar una señal.

Sin embargo, el componente más importante que hemos estudiado es el transistor. Éste es el componente esencial de todos los dispositivos que se utilizan para cambiar y manipular las señales en un ordenador. El transistor se puede emplear para amplificar una señal o para conmutar señales, encendiéndolas y apagándolas. Lo más importante, desde el punto de vista del ordenador, es que el transistor puede encender y apagar una señal de acuerdo a otra u otras señales. Esto es lo que sucede en el interior de una puerta lógica.

Hemos construido las tres puertas lógicas más sencillas, NOT, OR y AND, a partir de un puñado de componentes (aunque las puertas individuales del interior de los circuitos integrados del ordenador suelen estar construidas con otros tipos de transistores más complejos).

Las puertas lógicas no son particularmente útiles por sí mismas, pero se pueden combinar entre sí para formar circuitos lógicos que efectúen operaciones con los datos. En el capítulo anterior construimos un sumador incompleto, un circuito lógico sencillo para sumar dos números binarios, con las puertas lógicas de dos circuitos integrados.

Los circuitos integrados son de los componentes más complejos que se emplean en electrónica. Los chips simples de lógica transistor-transistor (*Transistor-Transistor Logic*: TTL) que ha utilizado hasta ahora son paquetes de integración a pequeña escala (*Small Scale Integration*: SSI). En cada chip hay solamente unos pocos transistores. Éste fue el primer tipo de chips que se fabricó y los primeros ordenadores se basaban en ellos. Sin embargo, a medida que la técnica se fue perfeccionando, aumentó el número de transistores que se podían incluir en un único chip. La integración a media escala (*Medium Scale Integration*: MSI) permitió disponer de circuitos lógicos completos en un chip.

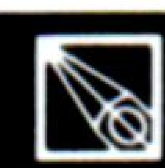
También hemos estudiado la integración a gran escala (*Large Scale Integration*: LSI) y la integración a muy gran escala (*Very Large Scale Integration*: VLSI). Una CPU completa en un único chip es un ejemplo de VLSI. Puesto que en el interior de un microprocesador hay miles de transistores, resulta difícil imaginar el circuito lógico de un chip de esta clase, pero esta complejidad hace que los chips individuales resulten muy eficaces y sencillos para el diseño de circuitos.

Una vez que dominemos estos fundamentos, podremos seguir adelante para abordar conceptos más complejos, lo que le brindará la posibilidad de construir algunas adiciones útiles para su micro. Primero, no obstante, ponga a prueba su destreza con los proyectos de la página contigua.

1) Resistencias

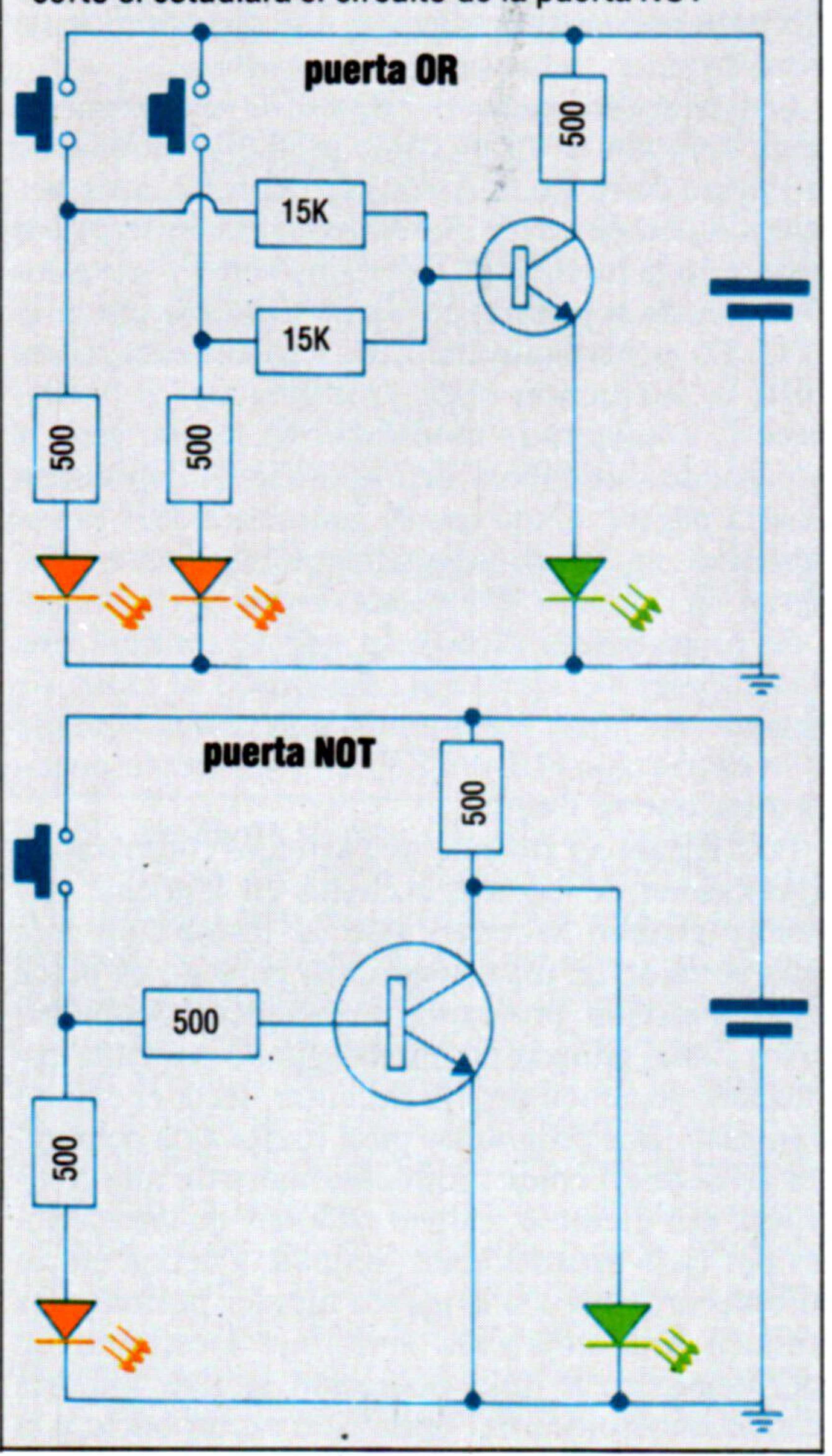
Las resistencias poseen una serie de franjas de colores para saber su valor. ¿Qué valores tienen estas resistencias? ¿Qué franjas tendría una resistencia de 150 ohmios?

a) b)



2) Puerta NOR

En la página 625 construimos puertas NOT, OR y AND empleando transistores. Este primer ejercicio práctico consiste en construir una puerta NOR utilizando un circuito similar a aquellos para NOT, OR y AND. Como ayuda, abajo le proporcionamos los circuitos para OR y NOT. Este problema se puede enfocar de dos maneras. Podría usar sus conocimientos sobre circuitos lógicos para construir una puerta NOR combinando una OR y una NOT. O bien, hallar un procedimiento más corto si estudiara el circuito de la puerta NOT



3) Conversor de decimal a binario

Construya un circuito que convierta de decimal a binario. Con el fin de que el circuito sea sencillo, restringiremos este ejercicio a números binarios de dos bits, es decir, los números decimales del cero al tres. Su circuito habrá de tener cuatro interruptores de entrada etiquetados cero, uno, dos y tres. Cuando pulse uno de estos interruptores, en un par de LED se deberán reflejar los dos bits binarios correspondientes. Una tabla de verdad parcial para el conversor sería así:

BOTÓN CERO	BOTÓN UNO	BOTÓN DOS	BOTÓN TRES	BIT HI	BIT LO
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

4) BCD, ¿sí o no?

En el curso sobre *Ciencia informática* hemos estudiado el sistema numérico decimal codificado en binario (BCD). Éste es un sistema de numeración a medio camino entre el decimal y el binario; cada dígito decimal se convierte a su equivalente binario y los grupos de cuatro bits resultantes se agrupan entre sí para formar el número BCD. Por ejemplo, 53 se codifica como 01010011, 5 se representa mediante 0101 y tres se representa mediante 0011. Esto significa que todo dígito BCD válido es un grupo de bits comprendido entre 0000 y 1001, correspondiente a entre 0 y 9 en decimal. En BCD son no válidos todos los códigos que no estén comprendidos dentro de la escala de 1010 a 1111.

Construya un circuito para comprobar si una entrada dada de cuatro bits es un dígito legal. Su circuito ha de tener cuatro interruptores: B0, B1, B2 y B3, que representan al número que se está probando. Deberá haber dos salidas: un LED verde si el número es un dígito BCD legal y, en caso contrario, un LED rojo. La tabla de verdad sería así:

Equivalente decimal	B3	B2	B1	B0	BCD válido	BCD no válido
0	0	0	0	0	1	0
1	0	0	0	1	1	0
2	0	0	1	0	1	0
3	0	0	1	1	1	0
4	0	1	0	0	1	0
5	0	1	0	1	1	0
6	0	1	1	0	1	0
7	0	1	1	1	1	0
8	1	0	0	0	1	0
9	1	0	0	1	1	0
10	1	0	1	0	0	1
11	1	0	1	1	0	1
12	1	1	0	0	0	1
13	1	1	0	1	0	1
14	1	1	1	0	0	1
15	1	1	1	1	0	1

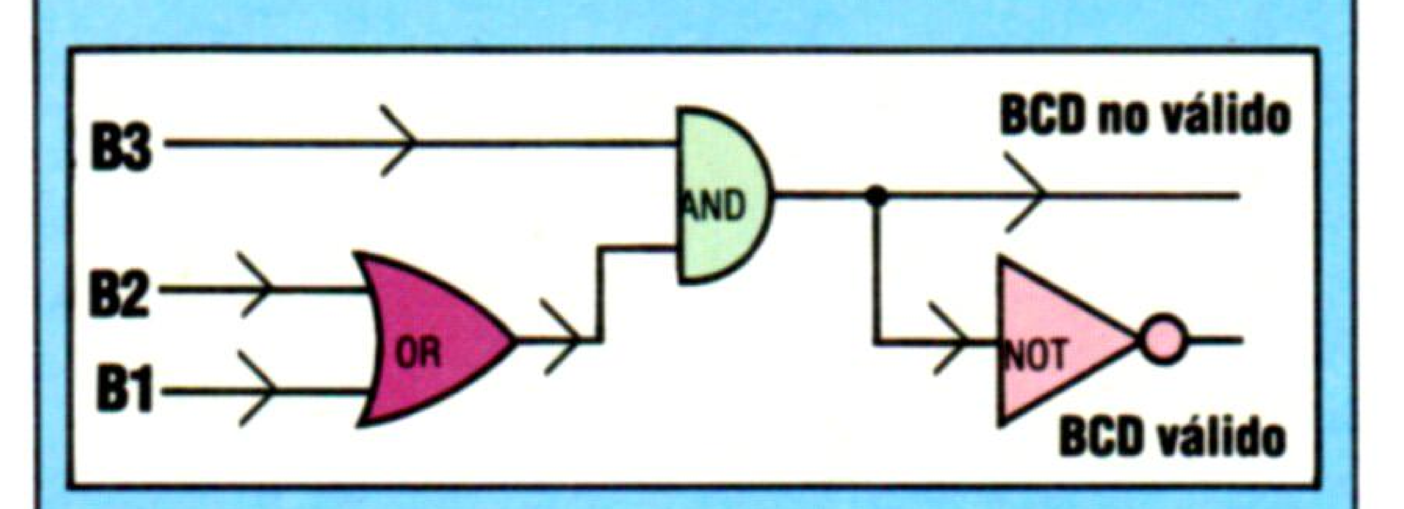
A partir de esta tabla de verdad podemos ver que la señal BCD viene dada por $\overline{B3} + \overline{B2}.\overline{B1}$. La señal BCD no válida es el NOT de ésta:

$$\overline{\overline{B3} + \overline{B2}.\overline{B1}}$$

que se puede simplificar así:

$$\overline{\overline{B3}}\,.\,\overline{\overline{B2}.\overline{B1}}$$
$$B3.(B2+B1)$$

De modo que el circuito para probar un dígito BCD no válido es bastante sencillo y sólo implica a tres de las entradas. Un diagrama lógico sugerido para el circuito convalidador BCD es:



Las respuestas, en el próximo capítulo de *Bricolaje*.

Recurso esencial

Ahora examinaremos un aspecto básico en la programación en lenguaje máquina: las modalidades de direccionamiento

Toda instrucción en lenguaje Assembly, explícita o implícitamente tiene que ver con los contenidos de la memoria, y puesto que un byte sólo se puede distinguir de otro por su dirección, toda instrucción en lenguaje Assembly debe relacionarse al menos con una dirección. El modo como se alude a una dirección puede ser directo y evidente, como en LDA $E349, que significa "cargar en el acumulador el contenido de la dirección $E349". En este caso, tanto el acumulador (un byte con nombre más que el número de su dirección) como la dirección $E349 se mencionan sin ningún tipo de ambigüedad y la naturaleza de la relación entre ambos está clara.

Pero la referencia a una dirección puede ser mucho menos obvia: RET, por ejemplo, que significa "retornar desde una subrutina". Ésta no parece referirse a ninguna dirección en concreto, mientras no entendamos que "la dirección de la siguiente instrucción a ejecutar se localiza allá donde se efectuó la llamada a la subrutina". Aquí, la dirección cuyo contenido se ha de cambiar (o sea, el contador del programa: el registro que retiene la dirección de la próxima instrucción a ejecutar) no se menciona, ni tampoco la dirección en donde se pueden encontrar sus nuevos contenidos (es decir, la nueva dirección de posición). Estas dos instrucciones se pueden considerar como ejemplos de *modalidades de direccionamiento* que contrastan entre sí.

Hasta el momento, en este curso hemos visto en las instrucciones dos tipos de modalidades de direccionamiento: *modalidad inmediata*, como en LDA A,$45 o ADC #$31, y *modalidad directa absoluta*, como en STA $58A7 o LD ($696C),A. Éstas podrían parecer las modalidades de direccionamiento "naturales", cubriendo todos los casos posibles, salvo las modalidades implícitas como RTS o RET, pero hay otras y vamos a analizarlas por separado.

Direccionamiento de página cero

El *direccionamiento de página cero* (también llamado *direccionamiento corto*) se utiliza siempre que una instrucción pida una dirección comprendida entre $0000 y $00FF. Todas las direcciones de este tramo poseen un byte *hi* (alto) de $00 y, por consiguiente, residen en la página cero de la memoria. Tales instrucciones sólo necesitan dos bytes: un byte para el opcode y otro para el byte *lo* (bajo) de la dirección. Cuando la CPU detecta una dirección de un solo byte en un punto donde espera que haya dos bytes, da por sentado un byte *hi* de $00 y, por tanto, entiende automáticamente la página cero. La ventaja de esta modalidad de página cero es la velocidad de ejecución: a los datos de la página cero se accede con una rapidez apreciablemente mayor que a los datos de cualquier otra página, precisamente porque requiere una dirección de un único byte.

Los microprocesadores Z80 y 6502 se diferencian sobre todo por la forma como usan esta modalidad de página cero. En el Assembly del 6502, toda instrucción con dirección RAM (como LDA) se puede utilizar en la modalidad de página cero, y todos los 256 bytes de la página cero están a disposición de la CPU. En el Assembly del Z80, sólo una instrucción (RST, la instrucción *reset*, "recomenzar" o "restablecer") configura la modalidad de página cero, y sólo puede direccionar ocho posiciones específicas de esta página. Dado que la instrucción RST es tan específica en su direccionamiento, requiere solamente un opcode (la dirección forma parte del opcode propiamente dicho), lo que le confiere una gran rapidez de ejecución. Avanzado el curso de lenguaje máquina, hablaremos más detenidamente de la instrucción RST del Z80, por los usos especiales a los que se destina.

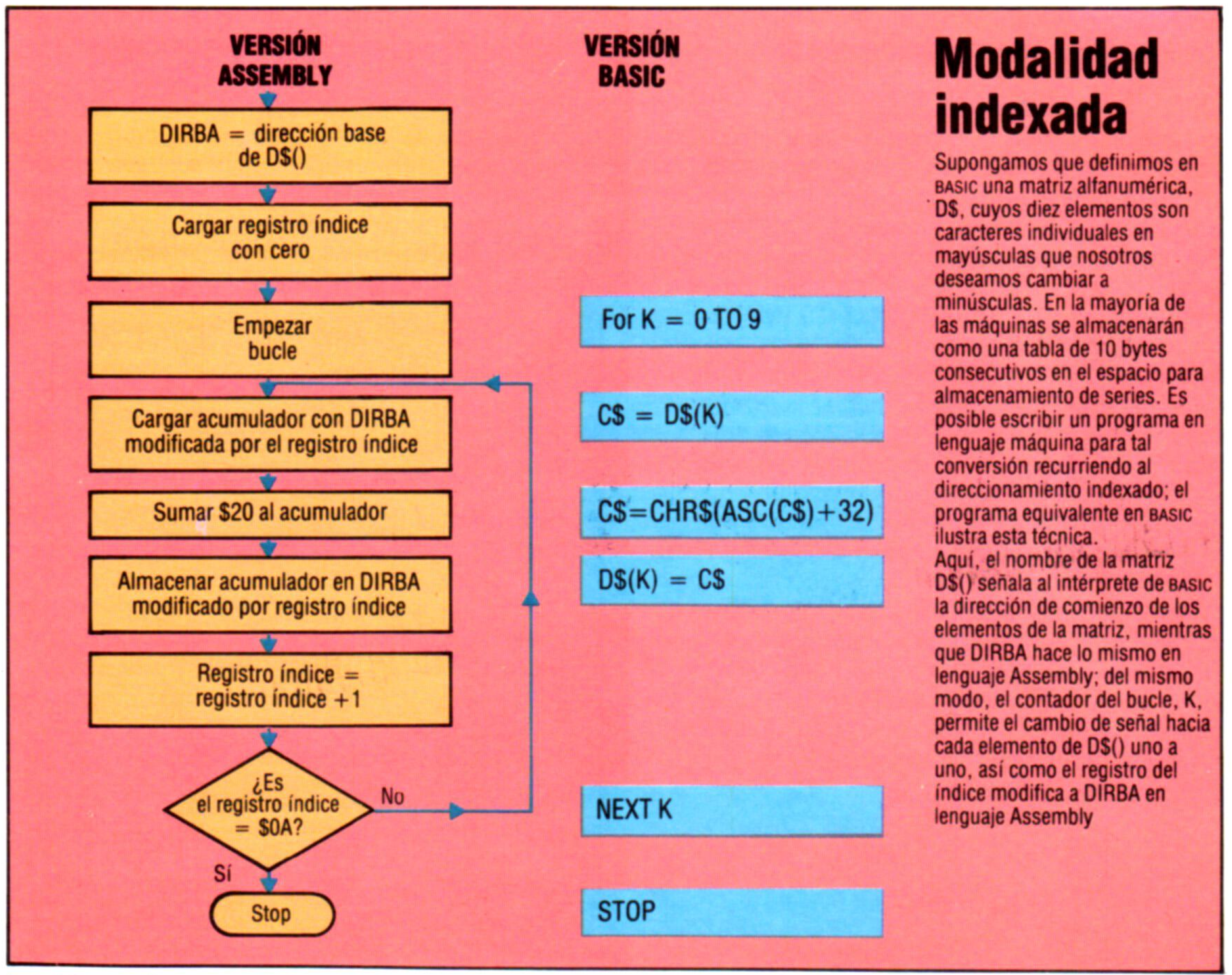
Podría parecer ridículo que estemos comparando la velocidad de las instrucciones en lenguaje Assembly cuando sabemos que la instrucción más lenta se mide en microsegundos; pero no es difícil llegar a escribir programas en lenguaje Assembly en los cuales ahorrar un microsegundo por cada instrucción ejecutada podría decantar hacia el éxito o el fracaso. Los programas para juegos que configuran gráficos en color tridimensionales de alta resolución, por ejemplo, exigen millones de instrucciones por cada operación de pantalla, y deben ejecutar estas órdenes con la mayor rapidez posible para producir una animación uniforme. Descontar un microsegundo de una operación sí que importa cuando uno coloca esa operación en un bucle y la ejecuta 64 000 veces de forma consecutiva.

Direccionamiento indexado

La *modalidad indexada* es fundamental para la programación en lenguaje Assembly, porque permite la construcción de estructuras de datos encadenados. Sin este tipo de estructuras los programas se limitan a dirigirse a posiciones de memoria individualmente por dirección: esto es lo que hemos venido haciendo hasta ahora en todos los programas explicados. Al disponer de la indexación, sin embargo, se pueden manipular más datos y se conceden mayores poderes al programador.

Los elementos esenciales de la modalidad indexada son una *dirección base* y un *índice*. Supongamos que queremos disponer de una tabla de datos (p. ej., los códigos de caracteres ASCII) en bytes consecutivos. La "direción base" de la tabla es la dirección de su primer byte. A partir de éste podemos aludir a cada uno de los bytes subsiguientes de





Modalidad indexada

Supongamos que definimos en BASIC una matriz alfanumérica, D$, cuyos diez elementos son caracteres individuales en mayúsculas que nosotros deseamos cambiar a minúsculas. En la mayoría de las máquinas se almacenarán como una tabla de 10 bytes consecutivos en el espacio para almacenamiento de series. Es posible escribir un programa en lenguaje máquina para tal conversión recurriendo al direccionamiento indexado; el programa equivalente en BASIC ilustra esta técnica.
Aquí, el nombre de la matriz D$() señala al intérprete de BASIC la dirección de comienzo de los elementos de la matriz, mientras que DIRBA hace lo mismo en lenguaje Assembly; del mismo modo, el contador del bucle, K, permite el cambio de señal hacia cada elemento de D$() uno a uno, así como el registro del índice modifica a DIRBA en lenguaje Assembly

Liz Dixon

la tabla mediante su posición relativa respecto a la dirección base, de modo tal que el primer byte está en la posición cero, el segundo byte está en la posición uno, el tercero en la posición dos, y así sucesivamente. La posición relativa de un byte respecto a la dirección base de la tabla es su "índice", y la dirección absoluta de cualquier byte de la tabla se calcula a partir de la suma de la dirección base y el índice del byte. Si pudiéramos construir un bucle de programas en lenguaje Assembly y utilizáramos el contador del bucle como un índice para la dirección base de la tabla, entonces obtendríamos la dirección de cada byte de la tabla en secuencia, de manera semejante a como accederíamos a los elementos de una matriz en BASIC mediante un bucle FOR ...NEXT.

Nuevamente, los lenguajes Assembly Z80 y 6502 tratan el direccionamiento indexado de forma diferente. El chip 6502 emplea dos registros de un solo byte, llamados X e Y, cada uno de los cuales puede contener un índice que modifique a una dirección base. Esto limita la longitud de una tabla a 256 bytes (el número máximo posible en un solo byte). El chip Z80 contiene dos registros de dos bytes, IX e IY, que pueden retener la propia dirección base y que pueden luego aumentar o disminuir para señalar a sucesivos bytes de la tabla. Puesto que son registros de dos bytes, IX e IY pueden direccionar tantos bytes como pueda direccionar la CPU.

Direccionamiento indirecto

El direccionamiento indirecto implica el uso de direcciones de señaladores, un concepto que ya habíamos introducido con anterioridad en el curso, en relación con las fronteras flotantes de la memoria (véase p. 538). Imaginemos que un grupo de personas forma un cine club y que están de acuerdo en ir una vez por semana a ver la película que recomienda el presidente del club. La película escogida puede que se esté exhibiendo en cualquiera de los doce cines distintos de la localidad, de modo que el presidente escribe semanalmente en una cartulina el título de la película, hora y lugar, y luego la pega en el escaparate convenido de una tienda del centro de la ciudad. Los socios del club no saben en qué sala se exhibirá la película semana tras semana, pero sí saben dónde se halla la tienda, y la tienda les "señala" el cine buscado. La dirección de la tienda es, indirectamente, la dirección de la sala de cine.



Ian McKinnell

Indicador indirecto
En nuestra vida cotidiana no son frecuentes los casos de direccionamiento indirecto. Sin embargo, en esta fotografía el panel de destino y número de andén de donde salen los trenes contiene la información que desean los viajeros, de modo que el rótulo colgante que les indica dónde hallar ese panel direcciona indirectamente los datos. En una instrucción Assembly, direccionamiento indirecto significa que en la dirección que indica el operando se encuentra la dirección del byte donde están almacenados los datos; la dirección del operando es un indicador

En el direccionamiento indirecto se pueden escribir instrucciones que contengan la dirección de un señalador y afecten al contenido de la posición que indique el señalador (no al contenido del señalador, por supuesto). Las ventajas que ofrece esta modalidad de direccionamiento son considerables, especialmente cuando se combina con la modalidad indexada. Supongamos, por ejemplo, que usted escribe una rutina en lenguaje Assembly para buscar una tabla de datos para un carácter determinado, y retorna con la posición del índice del carácter. Supongamos, además, que usted desea manejar varias tablas de este tipo situadas en distintos lugares de la memoria, y que desea emplear la misma rutina para buscar cualquiera de ellas. Si la rutina se escribiera de modo tal que encontrara la dirección base de la tabla de búsqueda indirectamente, a través de la posición del señalador escogido, entonces se podría utilizar cualquier tabla, siempre y cuando el contenido de la posición del señalador se ajuste convenientemente antes de llamar a la rutina.

En general, los programas requieren mezclas de estas modalidades y no ejemplos puros de modalidades únicas. La instrucción LDA del 6502, por ejemplo, se puede utilizar en las modalidades que ofrecemos en el cuadro.

Opcode	Operando	Modalidad
LDA	#$34	Modalidad inmediata
LDA	$A2	Directa página cero
LDA	$967F	Directa absoluta
LDA	$A2,X	Página cero, indexada por X
LDA	$967F,X	Absoluta, indexada con X
LDA	$967F,Y	Absoluta, indexada con Y
LDA	($A2,X)	Indirecta, preindexada con X
LDA	($A2),Y	Indirecta, postindexada con Y

En este ejemplo convenía utilizar una instrucción del 6502 porque ilustra con toda claridad las combinaciones de modalidades de direccionamiento. Observe que las dos versiones indirectas de la instrucción están en página cero, así como las modalidades indirectas e indexadas. Una tabla como ésta podría resultar confusa a primera vista, pero cuando se practican las diversas modalidades su significado queda claro enseguida. Hasta el momento usted ya conoce y emplea tanto LDA como ADC en dos modalidades (inmediata y absoluta) sin que haya ninguna confusión.

La tabla, además, responde a la pregunta que planteábamos en el capítulo anterior: ¿cómo distinguir la modalidad de direccionamiento de una instrucción cuando la mnemotécnica es la misma en todos los casos? Pues bien, se puede ver que el formato del operando es diferente en cada modalidad, y que la única ambigüedad posible es si una instrucción como LDA SYMB1 está en modalidad de página cero o absoluta. Un programa ensamblador le resolverá este problema, pero si es usted quien ensambla el programa "a mano", tendrá que verificar si SYMB1 se ha definido como una cantidad de un único byte o de dos bytes.

En general, una vez que se empieza a utilizar un ensamblador podemos olvidarnos de aquellas cosas como opcodes o número de bytes por instrucción, y concentrarnos en aprender las técnicas de programación del lenguaje Assembly. Es importante entender la mecánica del lenguaje máquina, pero el lenguaje Assembly utilizado con un buen ensamblador simbólico constituye una forma mucho mejor de programar, combinando el poder del lenguaje máquina con muchas de las facilidades de los lenguajes de alto nivel.

Respuestas al "Ejercicio" de p. 658

El programa monitor de la página 598 se escribió en módulos teniendo presente la idea de la ampliación, de modo que agregar una opción de menú es relativamente sencillo:

Versión para el Spectrum

1) Ajuste la inicialización editando o agregando las siguientes líneas:

```
1050 LET LT = 5:DIM C$(LT):DIM O$(LT,24):DIM
     X$(16)
1150 LET C$ = "ADGQB":LET C1 = -48:LET
     C2 = 10-CODE(C$(1))
1280 LET O$(5) = "   B-VISUALIZACIÓN EN
     BINARIO"
```

2) La rutina de entrada de la línea 2000 ya ha producido la dirección de comienzo, la ha normalizado como un número hexa de cuatro dígitos en A$, y la ha convertido en un número decimal en DN, y la subrutina para visualización en binario sería:

```
7000 REM**S/R VISUAL HEXA&BIN**
7020 FOR P = DN TO (DN + 15)
7040 LET NM = P:GOSUB 3100:PRINT H$,
7060 LET N = PEEK(P):LET NM = N
7080 GOSUB 3000: PRINT B$;"   ";
7100 GOSUB 7300: PRINT B$
7120 IF P = 65535 THEN LET P = DN + 15
7140 NEXT P
7200 RETURN
7300 REM **S/R BYTE BINARIO**
7310 LET B$ = ""
7320 FOR D = 8 TO 1 STEP-1
7330 LET N1 = INT(N/2)
7340 LET R = N-2*N1
7350 LET B$ = STR$(R) + B$
7360 LET N = N1
7370 NEXT D
7380 RETURN
```

Versión para el BBC y el Commodore

Copiar la versión del Spectrum, con las siguientes correcciones:

1) Cambiar la inicialización de LT y O$() como en la versión para el Spectrum, y agregar C$(5) = "B" en la línea 1150.

2) La línea 600 transfiere el control a la rutina de órdenes, por ello en el Commodore 64 y el BBC Micro debe cambiarse por:

```
600 ON CM GOSUB 5000,5500,6000,6500,7000
```

3) En el BBC, cambiar la línea 7060 de arriba por:

```
7060 N = ?(P):NM = N
```

4) En el Commodore 64 cambiar la línea 7350 anterior por:

```
7350 B$ = MID$(STR$(R),2) + B$
```



En constante avance

Hace cincuenta años era una pequeña firma proveedora de artículos eléctricos; hoy en día Tandy Corporation es una empresa multinacional sólida y de prestigio

Tandy Corporation, a través de sus departamentos de ventas al por menor, Tandy y la cadena norteamericana Radio Shack, posee 392 centros de ordenadores y más de 5 500 puntos de venta al detalle a lo largo de 76 países. La empresa cuenta con 29 fábricas que proveen equipos para la venta bajo las marcas comerciales Tandy y Radio Shack.

Tandy no comenzó su andadura como detallista de artículos electrónicos. La fundaron Norton Hinckley y David Tandy en 1927 como la Hinckley-Tandy Leather Company, una proveedora de cuero para las tiendas de reparación de calzado en Beaumont (Texas). El primer paso que la llevaría a convertirse en un gigante de la electrónica se dio en 1963, cuando Charles, el hijo de David Tandy, decidió ampliar el negocio y compró participaciones en una empresa con sede en Boston que se estaba tambaleando, denominada Radio Shack. Esta firma había venido operando desde los años veinte como un proveedor a pequeña escala de componentes eléctricos para radioaficionados y otros entusiastas de la electrónica. Aunque los negocios de la empresa se realizaban mayormente por pedidos por correspondencia y tenía un total de nueve tiendas en la zona de Boston, estaba experimentando enormes pérdidas. Para 1967 Charles Tandy había conseguido transformar los 4 millones de dólares de pérdidas en beneficios que ascendían a 20 millones.

El siguiente paso importante se dio en 1970, cuando Tandy se hizo cargo de una cadena de grandes almacenes, Leonards, que le dieron acceso al mercado de artículos eléctricos para el consumidor. Ahora esta área se ha desarrollado hasta el punto de que en el catálogo Tandy de 1984 se incluyen 2 625 artículos no relacionados con los ordenadores, desde resistencias a equipos de alta fidelidad y sintetizadores, y 396 productos de carácter informático.

Tandy llegó a Gran Bretaña en 1973 y se estableció rápidamente en los principales centros comerciales como detallista de artículos eléctricos. En 1978, cuando se lanzó en dicho país el microordenador TRS-80 Modelo I, Tandy contaba con 120 tiendas. Para 1983 el número se había elevado a 227 comercios detallistas extendidos a lo largo y a lo ancho del país. Veintiséis de éstos eran tiendas de informática, especializadas exclusivamente en ordenadores, software y periféricos.

El Modelo I hizo de Tandy uno de los más importantes fabricantes de ordenadores. Se trata de una máquina de teclado individual con un microprocesador Z80, al menos cuatro Kbytes de RAM y una pantalla en blanco y negro con gráficos de baja resolución. Se encuentran a la venta unidades de disco y los usuarios hasta pueden ejecutar el sistema operativo de gestión CP/M en su máquina.

Desde entonces la línea de Tandy siempre se ha mantenido actualizada, aunque nunca ha vuelto a reconquistar el dominio de mercado que le proporcionó el Modelo I. La empresa se introdujo rápidamente en el mercado del micro de oficina a través del Modelo II, y en la actualidad su gama incluye los Modelo 12 y Modelo 16, ambas máquinas de 16 bits, así como el nuevo Modelo 2000, compatible con el IBM-PC. La informática del Modelo I escaló posiciones en el mercado con el Modelo III, que se puede considerar como un ordenador personal caro o bien como una económica máquina de oficina. Tandy ha sustituido el III por el Modelo 4 (y su versión portátil, el 4P). Éstos poseen mejores facilidades de gestión, pero mantienen la compatibilidad con los programas para el Modelo I y el III: un



John Sayers, director de marketing en Gran Bretaña

Vince Moore, director de ventas en Gran Bretaña



Ian McKinnell

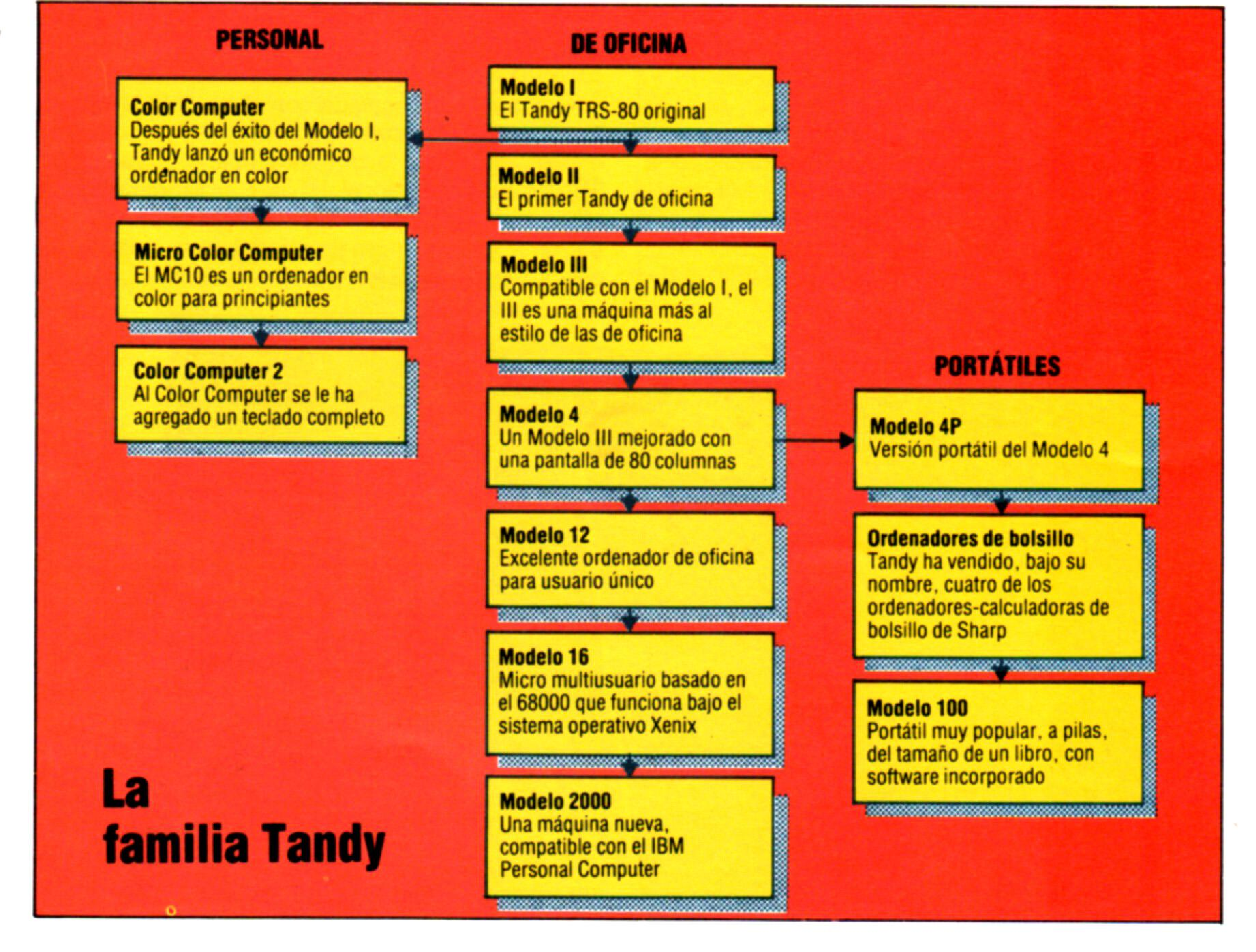
Por todo lo alto
La tienda más céntrica de Tandy realza el punto más fuerte de la empresa: un solo proveedor local de equipos, servicios y asesoramiento. En Estados Unidos el nombre Radio Shack de la empresa es igualmente conocido

logro poco habitual en la microinformática. El resultado de ello es que los modelos TRS-80 disponen de una amplia gama de software de apoyo.

Tandy intentó volver a establecerse en el mercado personal con el Tandy Color Computer, un micro basado en el 6809 que comparte muchas de las configuraciones del Dragon. Aunque en Gran Bretaña no ha alcanzado nunca un éxito notable, el "CoCo" se ha vendido bien en Estados Unidos. Pero Tandy no ha renunciado a esta área del mercado, como indica la existencia de una miniversión mejorada: el Micro Color Computer.

Quizá la línea más interesante haya sido la de los ordenadores portátiles. Tandy se dedicó intensamente a ella, vendiendo una serie de ordenadores de bolsillo basados en la gama Sharp. Más recientemente, un acuerdo con una empresa japonesa, Kyocera, y la firma de software norteamericana Microsoft, dio como resultado el Modelo 100, un portátil a pilas de reducidas proporciones (tiene el tamaño de un libro), pero muy completo, pues incorpora BASIC, procesador de textos, software para comunicaciones y un diario.

En 1984, varias de las grandes firmas de la industria del microordenador estaban empezando a experimentar cuantiosas pérdidas, pero los negocios de Tandy parecen saludables, con un flujo continuo de nuevos productos provenientes de sus centros de diseño de Fort Worth (Texas), donde la empresa tiene ahora su sede, y de su subsidiaria TC Electronics Corporation, instalada en Tokio. Con las ventajas que ofrecen un gran departamento de fabricación y su amplia red de comercialización al por menor, Tandy Corporation parece estar en condiciones inmejorables para sacar partido de todos los desarrollos, cualesquiera que sean, que pudieran producirse en el campo de la microelectrónica.



Editorial Delta, S.A.



10034
9 788485 822836